MSX DISK

A.C.J. Groeneveld

handboek voor iedereen



uw **MSX** *computer de baas*

MSX DISK

handboek voor iedereen

uw **MSX** *computer*
de baas

A.C.J. Groeneveld



uitgeverij STARK - TEXEL
postbus 302 - 1794 ZG Oosterend tel. 02223 - 661

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Groeneveld, A.C.J.

MSX disk handboek voor iedereen : uw MSX computer de baas / A.C.J. Groeneveld. – Oosterend : Stark-Texel
ISBN 90-6398-407-3
SISO 365.3 UDC 681.3.06
Trefw.: microcomputers ; programmeren.

....

1e druk 1985
ISBN 90 6398 407 3

INHOUD

1 INLEIDING

In het grote MSX-handboek werd de MSX-computer en het MSX-basic uitgebreid behandeld. De bezitter van een eenvoudige MSX-computer vindt in dit grote handboek dan ook alle geheimen van de MSX-computer uitgebreid beschreven.

Dit MSX-disk-handboek is bedoeld als een uitbreiding op dit eerste, grote handboek en is geschreven voor de bezitter van een MSX-computer met een schijveneenheid tezamen met het grote MSX-handboek of eventueel de verkorte versie van dit handboek die door veel importeurs standaard bij de computer wordt geleverd.

In dit handboek wordt met name ingegaan op het MSX-basic in verband met de schijveneenheid. De beginnende programmeur dient dan ook eerst een redelijke ervaring te hebben opgebouwd met behulp van het grote handboek alvorens dit handboek ter hand te nemen. Daarbij is het verstandig om het grote handboek altijd klaar te hebben liggen om nog eens het één en ander te kunnen nazoeken.

Tezamen met het grote MSX-handboek vormt dit MSX-disk-handboek een zeer duidelijke en complete handleiding voor uw MSX-computer met schijveneenheid.

Indien u nog niet in het bezit bent van het 'grote' handboek, dan kunt u dit alles onthullende, meer dan 400 pagina's dikke nederlandse handboek bestellen bij uw computerboekhandel. Uw boekhandelaar heeft aan het ISBN-nummer 90 6398 100 7 genoeg om het voor u te kunnen bestellen. Eventueel kunt u dit handboek direkt bij de uitgever bestellen (telefoon 02223 - 661).

Ik hoop dat dit MSX-disk-handboek het grote handboek in haar enorme sukses mag volgen.

april 1985,
A.C.J. Groeneveld.

2 MSX-BASIC EN MSX-DISK-BASIC

In het MSX-handboek werd reeds de geheugenopbouw van een MSX-computer behandeld. We zagen ondermeer dat we het geheugen van de MSX-computer kunnen indelen in vier geheugenbanken. In bank 0 en 1 zetelt het MSX-basic in ROM (read only memory, niet uitwisbaar geheugen). In bank 2 en 3 bevindt zich dan maximaal 32 kilobyte RAM (random access memory, vrij toegankelijk geheugen) waarin de MSX-computergebruiker bijvoorbeeld zijn programma kan koderen.

Het MSX-disk-basic brengt in de struktuur van de geheugenindeling enkele wijzigingen:

- allereerst resulteert het aansluiten van de disk-unit in een uitbreiding van het ROM-geheugen. 'Achter' geheugenbank 1 wordt een extra stuk ROM-geheugen geplaatst dat aktief wordt bij de benadering van de disk.
- ten tweede wordt een gedeelte van het RAM-geheugen 'afgesnoept' voor de disk. MSX-basic heeft bij het gebruik van een disk wat extra geheugen nodig voor het opslaan van tussenresultaten (bufferruimte).

Wanneer de disk verder niet wordt gebruikt, is de aanwezigheid van het disk-basic in plaats van het gewone basic alleen te merken aan de volgende verschijnselen:

- het beschikbare geheugen is wat kleiner. Sommige veel ruimte vragende programma's kunnen hierdoor een foutmelding geven in het MSX-disk-basic.
- indien bij de bevelen:

BLOAD	geen naam van het randapparaat wordt gegeven, werd
BSAVE	bij het MSX-basic automatisch aangenomen dat de
LOAD	cassetterecorder als randapparaat werd bedoeld. In
MERGE	het MSX-disk-basic wordt plotseling automatisch
OPEN	aangenomen dat de disk unit als randapparaat wordt
RUN	bedoeld.
SAVE	

- Het bevel SAVE "PROG" wordt in MSX-basic automatisch uitgevoerd op de cassetterecorder. In MSX-disk-basic wordt dit bevel echter automatisch op de disk uitgevoerd.

- Sommige fabrikanten leveren op floppy een kompleet disk operating system mee (MSX-DOS of CP/M). In dat geval dient het MSX-computersysteem te beschikken over 64 kilobyte RAM-geheugen. Voor het opstarten van het MSX-basic dient dan eerst het betreffende operating system te worden opgestart waarna MSX-basic via een bepaalde ingave dient te worden geaktiveerd. Raadpleeg in dat geval de beschrijving van het betreffende operating system.

- Sommige fabrikanten voorzien het MSX-disk-basic van een gewijzigde startmethode. Vaak dient voorafgaand aan het normale gebruik eerst een datum te worden ingegeven. Zie hiervoor de beschrijving van de fabrikant.

3 DE MSX-COMPUTER EN DE DISK

In het MSX-handboek werd reeds de opbouw van een MSX-computer behandeld. Eén van de onderdelen van de MSX-computer kan bestaan uit een (floppy) disk eenheid.

De disk eenheid wordt niet tot de standaard MSX-configuratie gerekend en werd als zodanig niet in het algemene handboek behandeld.

3.1 Permanente gegevensopslag

Het is bij een computer onontbeerlijk dat gegevens voor langere tijd kunnen worden bewaard. Een groot en ingewikkeld programma willen we niet elke avond opnieuw intikken, maar slechts éénmaal invoeren en vervolgens vastleggen. Standaard kan dit vastleggen van gegevens in MSX-basic gebeuren op een cassetteband met behulp van een cassette-recorder.

Elke MSX-programmeur stuit al snel tegen twee grote bezwaren van deze opslagmethode, namelijk:

- De opslag duurt vrij lang. 32 kilobytes (32768 tekens) op cassetteband vastleggen duurt ongeveer vijf minuten. Dit lijkt in eerste instantie snel, maar blijkt al vlug een groot bezwaar te zijn.

- De opslag is niet erg betrouwbaar. Vooral wanneer de cassette-recorder niet in topconditie is of verkeerd bandmateriaal wordt gebruikt, blijken gegevens tijdens opslag te kunnen worden verminkt. Het is dan ook noodzakelijk om uitstekend bandmateriaal aan te schaffen, de cassetterecorder regelmatig schoon te maken en af te stellen en de vastgelegde gegevens altijd te controleren.

Ondanks deze grote bezwaren is de cassetteband het meest voor de hand liggende opslagmedium voor hobby-computers; een cassetterecorder is relatief erg goedkoop, terwijl ook het bandmateriaal relatief zeer goedkoop is.

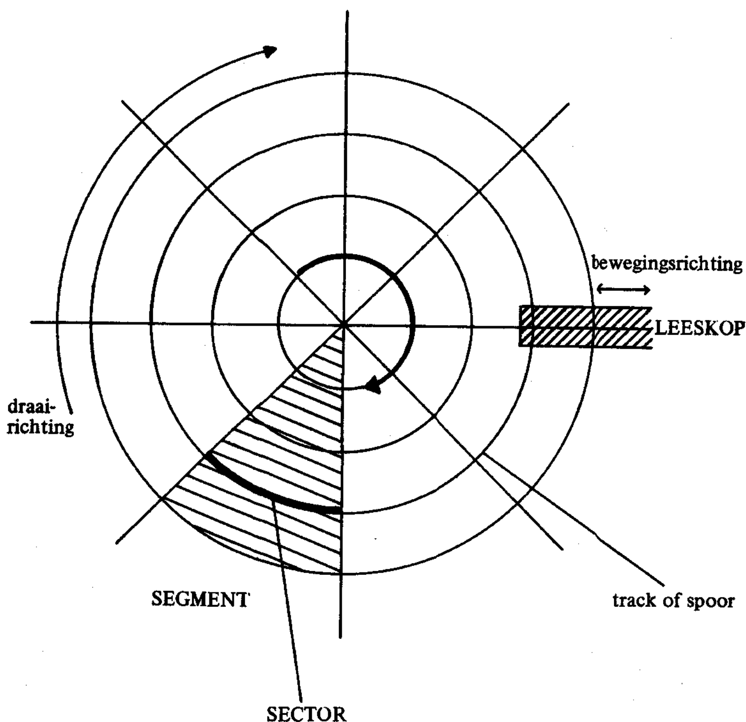
Wanneer snelheid bij het vastleggen en ophalen van gegevens een rol gaat spelen, wordt de cassetterecorder al snel een onbruikbaar randapparaat. Er dient dan naar een sneller opslagmedium te worden uitgekeken: de magneetschijf.

MSX-basic ondersteunt in een uitgebreide versie standaard het gebruik van deze magneetschijven.

In de volgende paragrafen wordt eerst het algemene principe van de magneetschijf uitgelegd. Daarna gaan we in op de logische indeling van een magneetschijf. Daarna, in hoofdstuk 4, wordt het MSX-basic met betrekking tot de magneetschijf behandeld.

3.2 De magneetschijf

De magneetschijfeenheid (disk unit) gaat uit van het principe dat gegevens worden gelezen van en geschreven naar een snel ronddraaiende schijf. Deze schijf, van aluminium of kunststof, is bedekt met een magnetisch geprepareerde laag, die vergelijkbaar is met de laag magnetisch materiaal die op een cassetteband is aangebracht.



We kunnen een magneetschijf ingedeeld zien in een aantal 'taartpunten' die we segmenten noemen. Daarbij kunnen we ons allemaal om elkaar heen liggende cirkels op het schijfoppervlak voorstellen, de sporen of tracks. De doorsnede van een segment met een track noemt men een sector. We kunnen een magneetschijf dus onderverdeeld zien in een groot aantal sectoren, verdeeld over de verschillende tracks op het schijfoppervlak.

Een sector is de eenheid van uitwisseling met een schijf. Op een sector kan een blok gegevens, meestal ter grootte van 512 of 1024 bytes, in één keer worden geschreven. Ook wordt een blok altijd in één keer ingelezen.

Om gegevens uit een bepaalde sector te kunnen lezen of om een bepaalde sector te kunnen (her) schrijven, is een lees-schrijfkop nodig, net zoals een opname- en weergavekop in een cassetterecorder noodzakelijk zijn. In het geval van de magneetschijf zit de gekombineerde lees-schrijfkop op een beweegbare arm gemonteerd. Deze arm wordt door een speciale motor naar binnen of naar buiten bewogen, zodat de leesschrijfkop alle tracks kan bereiken.

Magneetschijven kunnen aan één kant (single sided) of aan twee kanten (double sided) worden beschreven, afhankelijk van de schijveneenheid. Wanneer een magneetschijf aan twee zijden beschreven wordt, zijn er ook twee lees/schrijfkoppen noodzakelijk; één voor elke kant.

Om gegevens uit een bepaalde sector te lezen, dient de schijveneenheid de volgende akties te ondernemen:

1) beweeg de arm zo, dat de kop boven de juiste track wordt geplaatst.
2) wacht totdat de juiste sector voorbij draait.
3) lees de gegevens van deze sector in in het computergeheugen.

Om gegevens op schijf vast te leggen, dienen de volgende akties te worden ondernomen:

1) beweeg de arm zo, dat de kop boven de juiste track wordt geplaatst.
2) wacht totdat de juiste sector voorbijdraait.
3) schrijf de gegevens vanuit het computergeheugen in die sector.

Enkele voordelen van de magneetschijf ten opzichte van de cassetterecorder vallen onmiddellijk op:

– Er behoeft geen cassetterecorder meer op opnemen of afspelen te worden gezet; de schijveneenheid zorgt zelf voor in- en uitschakeling voor 'opnemen' en 'afspelen' van gegevens.

– Alle sectoren zijn in onderdelen van seconden te bereiken; lange door-of terugspoeltijden komen hierdoor te vervallen.

– De schijveneenheid is speciaal voor gegevensopslag ontworpen. In tegenstelling tot een gewone cassetterecorder behoeft de schijveneenheid niet geschikt te zijn om muziek te registreren. Hierdoor kan de kwaliteit van een disk unit worden toegespitst op de opslag van gegevens hetgeen een sneller en veel betrouwbaarder opslaan van gegevens mogelijk maakt.

3.3 Een snelheidsvergelijking

Voor diegene die hierin is geïnteresseerd, volgt hieronder een globale snelheidsvergelijking tussen een disk unit en een cassetterecorder.

DE CASSETTERECORDER

Om 16 kilobytes op cassetteband vast te leggen, is ongeveer de volgende tijd benodigd:

De hoogste schrijfsnelheid (onbetrouwbaar) op cassette is 2400 baud hetgeen overeenkomt met ongeveer 250 bytes per seconde. Om 16 kilobytes aan gegevens op band op te slaan, is dus een tijd nodig van ongeveer:

$$\frac{16 \times 1024}{250} = \text{66 seconden (ruim 1 minuut)}$$

Indien in een sector die 512 bytes kan bevatten, er 9 sectors per track zijn en de rotatiesnelheid van de disk ongeveer 5 maal per seconde is (allemaal redelijke aannamen voor een eenvoudige floppy disk eenheid) dan duurt het schrijven van 16 kilobytes aan gegevens ongeveer:

$$\frac{16 \times 1024}{512 \times 9 \times 5} = \text{nog geen seconde...}$$

Beide tijdsberekeningen zijn slechts benaderingen; sommige invloeden zijn volledig verwaarloosd. Zonder veel gevaar kan men echter stellen dat de eenvoudigste schijveneenheid al snel HONDERD MAAL ZO SNEL is als een cassetterecorder...

3.4 Vormen van magneetschijven

De magneetschijfeenheid of disk unit komt in diverse verschillende vormen voor. Drie vormen van magneetschijfeenheden behandelen we hieronder:

- de 3½ inch floppy disk unit

  Deze schijveneenheid maakt gebruik van 3,5 inch floppy disks. Deze floppy disks hebben een doorsnede van 3,5 inch (ongeveer 9 cm) en zijn verpakt in een hard plastic omhulsel. Het leesvenster is meestal afgedekt met een klepje en schuift bij het insteken van de floppy in de schijveneenheid automatisch open, waarna de lees/schrijfkop toegang heeft tot de daadwerkelijke schijf. De schijf zelf is gemaakt van een flexibele kunststof met daarop een magnetisch geprepareerde laag.
  Over het algemeen kent deze schijf-vorm een redelijk hoge opslag; per floppy disk kan meestal een hoeveelheid van 250.000 tot 500.000 tekens worden opgeslagen...

- de 5¼ inch floppy disk unit

  Deze schijveneenheid maakt gebruik van grotere floppy disks (5½ inch = ongeveer 13 cm doorsnede). Deze floppy disks hebben een open leesvenster en moeten daarom altijd in een speciale envelop worden bewaard. De verdere specificaties zijn ongeveer gelijk aan die van de 3½ inch floppy disk met dit verschil dat de opslagcapaciteit meestal wat hoger ligt (tot maximaal ongeveer 1.000.000 tekens).

- de winchester unit (hard disk)

  De schijf in een winchester unit kan niet worden uitgenomen. Meestal is een winchester disk gemaakt van een hard materiaal (aluminium) waaraan dan ook de benaming 'hard disk' is ontleend. Doordat de schijf vast in de schijveneenheid is opgenomen en daarbij van niet flexibel materiaal is gemaakt, is het voor de fabri-

kant mogelijk om de schijf harder te laten draaien en nauwkeuriger af te stellen. Grotere benaderingssnelheden en grotere opslagmogelijkheden zijn hiervan het gevolg. De gemiddelde winchester is al gauw weer 50 maal zo snel als een floppy en kan meestal 5 tot 10 miljoen tekens bergen...
Door de hoge prijsstelling is een harde schijf meestal niet voor de hobbyist weggelegd.

3.5 Write protect

Bij elke vorm van magneetschijven is de mogelijkheid aanwezig om de gehele schijf te beschermen tegen abusievelijk overschrijven van gegevens. Deze mogelijkheid noemt men meestal de WRITE PROTECT optie.

De 3½ diskette bezit hiertoe in één van de hoeken een klein schuifje. Door dit schuifje zo te zetten dat een gaatje ontstaat, wordt het schijfje beschermd tegen het per ongeluk overschrijven van gegevens. Nadat het schuifje op beschreven wijze in de write protect stand is geschoven, is het voor de computer onmogelijk om nog gegevens op de schijf te schrijven. Het blijft echter mogelijk om gegevens van deze schijf te LEZEN.

De 5¼ inch floppy disk heeft geen schuifje. In plaats hiervan dient een plakkertje te worden geplakt over de kleine insparing die zich aan de kant van het leesvenster bevindt. Nadat dit plakkertje is aangebracht, is het voor de computer niet meer mogelijk om nog op de schijf te schrijven. LEZEN gaat natuurlijk nog wel.

De hard disk unit bezit meestal een write protect knopje waarmee de schijf onbeschrijfbaar kan worden gemaakt.

Bij optredende fouten bij het schrijven geldt altijd dat u eerst dient te controleren of de schijf niet via de WRITE PROTECT optie is beschermd.

Wanneer dit niet het geval blijkt te zijn, onderzoek dan of de schijf wel geformatteerd is (zie de volgende paragraaf).

3.6 Formatteren

We zagen reeds dat een magneetschijf is onderverdeeld in zogenaamde sectoren. Elke sector staat op een vaste plaats op de magneetschijf. In elke sector kan een blok gegevens (meestal 512 of 1024 bytes) worden geschreven.

Voordat een magneetschijf (bijvoorbeeld een floppy disk) in gebruik wordt genomen, dient de onderverdeling in sectoren eerst te worden aangebracht op de schijf. De magneetschijf moet als het ware voor het gebruik eerst in stukjes worden verdeeld.

Het verdelen van de magneetschijf in partjes noemt men het FORMATTEREN van de magneetschijf. In de meeste gevallen geldt:

EEN SCHIJF MOET EERST WORDEN GEFORMATTEERD, EERDER KAN HIJ NIET WORDEN GEBRUIKT.

Daar de vorm van de magneetschijveneenheid door MSX niet wordt vastgelegd, is er ook geen vast omschreven manier van formatteren van een magneetschijf. Het formatteren van een schijf is als kommando niet in de MSX-standaard opgenomen.

Hoe een magneetschijf moet worden geformatteerd, dient in de gebruiksaanwijzing van de betreffende schijveneenheid te worden opgezocht. Meestal gebeurt dit door middel van een ingave als:

```
CALL FORMAT.
```

Hierna vraagt de computer dan welke schijf dient te worden geformatteerd. Geef de letter A in indien u maar één schijveneenheid heeft. Heeft u twee schijveneenheden, geef dan een A in voor de eerste of een B in voor de tweede schijveneenheid.

Meestal duurt het formatteren één tot twee minuten. Na afloop van het formatteren kan het zijn dat de computer een foutmelding geeft. Probeer in dat geval een tweede formattering. Blijft het formatteren fout gaan, dan is de schijf kapot of is de schijveneenheid vervuild of kapot.

PAS OP: WANNEER U EEN VOORBEELDSCHIJFJE MET EEN OPERATING SYSTEM MEEGELEVERD HEBT GEKREGEN, FORMATTEERT U DEZE NATUURLIJK NIET. MET HET FOR-

MATTEREN VERWIJDERT U NAMELIJK ALLE GEGEVENS VAN DE SCHIJF.

Ook indien u een gehele schijf in één klap leeg wilt hebben, kunt u dit door het formatteren bewerkstelligen.

Voordat u verder leest is het zaak dat u, wanneer u tenminste de voorbeelden uit wilt proberen, weet hoe u moet formatteren. Houd één geformatteerde schijf apart voor het oefenen met de voorbeelden.

3.7 De logische indeling van een magneetschijf

In de vorige paragrafen stelden we ondermeer vast dat we de magneetschijf uiteindelijk kunnen zien ingedeeld in een aantal sectoren. Elke sector bevat een blok van gegevens ter grootte van 512 of 1024 bytes. Zo'n blok is de kleinste eenheid die in zijn geheel kan worden gelezen of geschreven.

De indeling van de schijf in sporen, segmenten en sectoren (blokken) noemen we de fysische indeling van de schijf. Deze indeling kan als het ware op de schijf zelf worden aangewezen.

Met de logische indeling van de schijf bedoelen we de indeling die het MSX-basic hanteert, gebruik makende van de fysische indeling.

Doordat MSX-basic een intelligente en doordachte logische indeling van de magneetschijf ondersteunt, behoeft de gebruiker van de MSX-computer nooit stil te staan bij de fysische indeling van de magneetschijf. MSX bepaalt de fysische indeling, zorgt dat een blok niet ten onrechte door een ander kan worden overschreven en houdt bij in welke blokken welke gegevens zijn geschreven en welke blokken bij elkaar horen. Nooit behoeft de gebruiker dus een lijstje bij te houden van welke sectoren er werden gebruikt voor opslag van een programma. Door alleen maar de naam van het programma aan het MSX-systeem te verschaffen, kan worden bewerkstelligd dat de sectoren waarin het programma is opgenomen in de juiste volgorde worden ingelezen in het computergeheugen.

MSX-basic zorgt ervoor dat op elke disk een *inhoudsopgave* (index) wordt bijgehouden. In deze inhoudsopgave staat vermeld:

– wat de naam is van de verzameling van gegevens die op schijf werd

geschreven.
- wat de soort is van deze verzameling van gegevens. Zijn het programmaregels of zijn het andere gegevens?
- welke sectoren worden bezet door deze gegevens. Totdat deze gegevens worden verwijderd, mogen de sectoren, waarin deze gegevens staan, niet voor andere doeleinden worden gebruikt. MSX-basic waakt hierover.

Een bij elkaar behorende verzameling van gegevens noemt men in de computerwereld over het algemeen een BESTAND of een FILE. De MSX-schijveneenheid is een bestandgeoriënteerd medium. Voordat gegevens op floppy worden geschreven, dient eerst een bestand te zijn gedefinieerd op schijf. Hierop gaan we bij de behandeling van de kommando's verder in.

MSX verzorgt de besturing van de bestanden (het file management) waardoor we als MSX-programmeur niet met de fysieke opbouw van de schijveneenheid worden gekonfronteerd.

3.8 FILES

Zoals in de vorige paragraaf werd opgemerkt, gaat MSX-basic uit van een bestandsstruktuur op schijf. De term FILE of BESTAND zal steeds weer terugkeren en is zeer belangrijk. Onthoudt:

EEN BESTAND OF EEN FILE IS EEN GROEP VAN BIJ ELKAAR BEHORENDE GEGEVENS.

We onderscheiden twee verschillende soorten FILES, namelijk:

1. de PROGRAM FILE (programmabestand). Zo'n programmabestand bestaat uit allemaal bij elkaar behorende programmaregels die tezamen een programma vormen. Een program file wordt met behulp van het SAVE-kommando op de schijf geschreven en met behulp van het LOAD-kommando weer in zijn geheel van schijf opgehaald. Met een KILL-kommando kan een program file definitief van de schijf worden gewist.

2. de DATA FILE (gegevensbestand). Een data file bestaat uit allemaal bij elkaar behorende gegevens. In een data file kunnen we bijvoorbeeld de namen, adressen en telefoonnummers van onze kennissen opslaan. Ook kunnen we een ander bestand ontwerpen

met daarin de gegevens van onze postzegelverzameling, platenverzameling etcetera. Al deze bestanden mogen eventueel op één enkele disk staan; MSX-basic zorgt ervoor dat deze gegevens uit elkaar worden gehouden.

3.9 PROGRAM FILES

De PROGRAM FILE bestaat uit een verzameling van bijekaar behorende programmaregels die tezamen een programma vormen. Laten we als voorbeeld eens het volgende programma nemen:

```
NEW
Ok
10 PRINT "MSX-BASIC *** ";
20 GOTO 10
SAVE "PROG"
Ok
FILES
EDIT              PROG
KLARA
Ok
LOAD "PROG"
Ok
KILL "PROG"
Ok
FILES
EDIT              KLARA
Ok
```

In dit voorbeeld werd eerst een NEW-kommando gegeven waarna een programma van twee regels werd ingetikt. Hierna werd het programma met behulp van een SAVE-kommando op schijf gezet. Vervolgens werd het kommando FILES ingetikt; de computer geeft aan welke files nu op de schijf staan. Behalve de file PROG (het zojuist op schijf gezette programma) bleken er nog twee files op de schijf te staan.
Vervolgens werd het zojuist op schijf gezette programma weer in het computergeheugen ingelezen met behulp van het LOAD-kommando. Met het KILL-kommando werd vervolgens het programma PROG

weer van schijf verwijderd. Het laatste FILES-kommando laat zien dat de program file PROG daadwerkelijk van schijf is verdwenen.

Bovenstaand voorbeeld vertoont veel gelijkenis met de wijze waarop op cassetteband een programma kan worden opgeslagen. Over het algemeen wordt een program file met de sleutelwoorden SAVE, LOAD, KILL en MERGE benaderd.
Op deze sleutelwoorden gaan we in hoofdstuk 4 verder in.

3.10 DATA FILES

Data files of gegevensbestanden kunnen we weer in twee verschillende groepen onderverdelen:

1 de SEQUENTIAL FILE (sequentieel=in volgorde)

2 de RANDOM FILE (random=willekeurig)

De sequential file kan alleen in opklimmende volgorde worden benaderd. Gegevens die in een bepaalde volgorde op schijf werden gezet, kunnen alleen ook weer in deze volgorde worden ingelezen.

De random file geeft de schijveneenheid zijn grote kracht. De random file staat toe dat gegevens in elke volgorde kan worden benaderd. Wanneer we bijvoorbeeld onze kennissen nummeren van 1 tot 100 en deze vervolgens op schijf zetten, is het bij de random file niet noodzakelijk dat eerst de gegevens van kennis 1 tot en met 99 worden ingelezen om uiteindelijk de gegevens van kennis 100 te weten te komen. De gegevens van kennis nummer 100 kunnen onmiddellijk in het computergeheugen worden ingelezen.

3.11 SEQUENTIAL FILES

De sequential file geeft de mogelijkheid om gegevens achtereenvolgend op schijf te zetten. Nadat deze gegevens op schijf zijn gezet, kunnen ze ook weer in het computergeheugen worden ingelezen, echter alleen in die volgorde waarin ze ook geschreven zijn.

Gegevensbestanden op cassetteband kunnen eveneens worden beschouwd als sequentiele bestanden. Ook van cassetteband kunnen gegevens alleen in die volgorde worden gelezen waarin ze ook geschreven

zijn. Het voordeel van een sequential file op schijf is gelegen in snelheid en betrouwbaarheid. Gegevens worden vaak 100 maal zo snel van magneetschijf ingelezen, terwijl de integriteit (betrouwbaarheid) van deze gegevens bijzonder groot is.

Een voorbeeld:

```
NEW
Ok
10 OPEN "TEST" FOR OUTPUT AS #1
20 LINE INPUT R$
30 IF R$="" THEN 50
40 PRINT #1,R$:GOTO 20
50 CLOSE:OPEN "TEST" FOR INPUT AS #1
60 IF EOF(1)=-1 THEN CLOSE:STOP
70 INPUT #1,R$:PRINT R$:GOTO 60
RUN
DEZE TEKST WORDT IN EEN SEQUENTIAL
FILE GESCHREVEN EN ZO METEEN WEER
AFGEDRUKT.

DEZE TEKST WORDT IN EEN SEQUENTIAL
FILE GESCHREVEN EN ZO METEEN WEER
AFGEDRUKT.
Break in 60
Ok
```

Op regel 10 wordt een file aangemaakt, genaamd TEST. FOR OUTPUT wil zeggen dat gegevens naar deze file *geschreven* dienen te worden. Op regel 20 kan een tekst worden ingegeven. Wanneer daadwerkelijk een tekst werd ingegeven, volgt op regel 40 het kommando om deze tekst in de file te schrijven. Vervolgens kan dan een nieuwe regel tekst (GOTO 20) worden ingegeven.
Wordt op regel 30 gekonstateerd dat er geen tekst meer werd ingevoerd, dan wordt op regel 50 de file eerst gesloten en dan weer geopend, dit maal FOR INPUT. FOR INPUT wil zeggen dat gegevens vanuit de file dienen te worden *ingelezen.* Wanneer EOF(1) ongelijk is aan −1, dus wanneer nog geen einde bestand werd gekonstateerd, wordt op regel 70 een volgende regel aan gegevens ingelezen en geprojekteerd op het

beeldscherm. Indien op regel 60 een einde bestand werd gekonstateerd, wordt het bestand gesloten en is het programma ten einde.

Merk op dat het programma in het geheel geen nieuwe kommando's bevat. Wanneer geen schijveneenheid zou zijn aangesloten, zou hetzelfde programma op band zijn werk doen. Het is dan echter wel noodzakelijk om de cassette-recorder eerst op opnemen te zetten, vervolgens op terugspoelen en uiteindelijk op afspelen.

Het is erg leerzaam om dit programma een keer op cassette en een keer op magneetschijf uit te proberen; het snelheidsverschil is verbazend...

Het bovenstaande voorbeeld bestuurt een sequentieel bestand. De gegevens die eerst werden geschreven, werden daarna noodzakelijkerwijs in dezelfde volgorde weer ingelezen.

3.12 RANDOM FILES

Om het hoe en waarom van random files goed te begrijpen, is een nadere uitleg noodzakelijk.

We zagen reeds dat we op een magneetschijf meerdere files kunnen opslaan. In feite kunnen we de magneetschijf opgedeeld zien in FILES. Zo'n file wordt gevormd door een groep van bij elkaar behorende gegevens.

Een file kunnen we weer opgedeeld zien in kleinere stukken; de RECORDS.

EEN RECORD IS EEN EENHEID VAN GEGEVENS BINNEN EEN FILE. MEESTAL BEVAT EEN RECORD DE GEGEVENS VAN ééN OBJEKT, PERSOON OF VERSCHIJNSEL BINNEN DE FILE WAARIN DE GEGEVENS VAN MEERDERE OBJEKTEN, PERSONEN OF VERSCHIJNSELEN ZIJN OPGENOMEN.

Een voorbeeld: we leggen een bestand aan met daarin verschillende gegevens van onze vrienden en kennissen. We spreken af dat elke kennis een vast nummer van 1 tot 100 krijgt.

Per kennis willen we vastleggen:

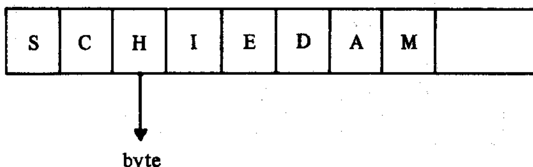
1. Zijn/haar nummer
2. Zijn/haar naam
3. Zijn/haar straat
4. Zijn/haar woonplaats
5. Zijn/haar telefoonnummer

Het bestand, dat we KENNIS zullen noemen, bevat dus de gegevens van al onze vrienden en kennissen. Eén record uit het bestand bevat het nummer, de naam, de straat, de woonplaats en het telefoonnummer van één kennis. Het bestand bestaat dus uit diverse records terwijl elk record bij precies één kennis hoort.

Een record kunnen we weer verdeeld zien in verschillende FIELDS of velden. In één veld wordt een volledige gegevenseenheid opgeslagen. Zo vormt de naam of de woonplaats één van de velden binnen het record van een kennis.

Ter verduidelijking eerst een schema:

FIELD:



In het bovenstaande schema zien we ons kennissenbestand weer terug. We zien dat we per kennis één record hebben gereserveerd. Per record kennen we weer enkele velden. In het voorbeeld lieten we eerst het record van kennis nummer 3 zien (Piet Jansen). Vervolgens beschouwden we het veld van de woonplaats binnen dit record (Schiedam).

Uiteindelijk bestaat een veld natuurlijk uit bytes; de kleinste eenheid van gegevensopslag waarin één enkel letterteken kan worden gekodeerd.

Hieronder volgt een voorbeeld van de besturing van een random file in MSX-basic.

```
NEW
Ok
10 OPEN "KENNIS" AS #1
20 FIELD #1,1 AS MM$,24 AS NN$,24 AS SS$
,24 AS WW$,24 AS TT$
30 PUT #1,101
40 CLS:PRINT "VULLEN KENNISSENBESTAND"
50 LOCATE 0,3:INPUT "KENNISNUMMER";K
60 IF K=0 THEN CLOSE:STOP
70 IF K>100 THEN BEEP:GOTO 40
80 IF K<>INT(K) THEN BEEP:GOTO 40
90 GET #1,K
100 IF MM$<>"*" THEN LSET MM$="*":LSET N
N$=" ":LSET SS$=" ":LSET WW$=" ":LSET TT
$=" "
110 N$=NN$
120 S$=SS$
130 W$=WW$
```

```
140 T$=TT$
150 LOCATE 0,5
160 PRINT "NAAM? ";N$
170 PRINT "STRAAT? ";S$
180 PRINT "WOONPLAATS? ";W$
190 PRINT "TELEFOONNUMMER? ";T$
200 LOCATE 0,5
210 INPUT "NAAM";N$
220 INPUT "STRAAT";S$
230 INPUT "WOONPLAATS";W$
240 INPUT "TELEFOONNUMMER";T$
250 LSET NN$=N$:LSET SS$=S$
260 LSET WW$=W$:LSET TT$=T$
270 PUT #1,K
280 GOTO 40
RUN


(beeld wordt schoongemaakt)


VULLEN KENNISSENBESTAND


KENNISNUMMER? 3

NAAM? PIET JANSSEN
STRAAT? STRAATWEG 11
WOONPLAATS? SCHIEDAM
TELEFOONNUMMER? 010-125684
```

Met dit programma kunnen honderd verschillende kennissen op de schijf worden opgeslagen. Indien een kennis reeds is ingebracht en zijn of haar nummer een tweede maal wordt ingegeven, verschijnen de gegevens weer op het beeldscherm. Deze gegevens kunnen eventueel worden verbeterd of met alleen een return onveranderd worden opgenomen.

Merk op dat de kennissen door elkaar heen kunnen worden aangemaakt en ook weer door elkaar heen worden opgevraagd. Er is geen verplichting tot het handhaven van enige volgorde.

Het bovenstaande programma behandelen we hier globaal; voor de preciese betekenis van de kommando's dient hoofdstuk 4 te worden geraadpleegd.

Op regel 10 wordt het kennissenbestand geopend en eventueel aangemaakt als het nog niet eerder bestond. Vervolgens wordt op regel 20 met het FIELD-kommando de indeling van het record bepaald. Eén karakter, MM$, wordt gereserveerd als merkteken. Bij een gevuld record wordt deze MM$ op een sterretje gezet als teken dat er goede informatie in het record aanwezig is. De velden NN$, SS$, WW$ en TT$ zijn elk 24 posities lang en zijn bedoeld voor de opslag van naam, straat, woonplaats en telefoonnummer.

Op regel 30 wordt vervolgens record nummer 101 beschreven met behulp van het PUT-kommando. Omdat NN$, SS$, WW$ en TT$ nog niet werden gevuld, wordt een leeg record geschreven op positie nummer 101 in het kennissenbestand. Dit record wordt bij het begin geschreven om te voorkomen dat later een input past end foutmelding verschijnt. Deze fout kan natuurlijk ook met de ON ERROR GOTO-konstruktie worden opgevangen. Echter, we houden het voorbeeld hier wat eenvoudig.

Op regel 40 wordt het beeldscherm schoongemaakt en een kopregel afgedrukt. Op regel 50 kan dan een kennisnummer worden ingegeven dat op regel 60, 70 en 80 wordt gekontroleerd op juistheid. Het programma stopt indien een kennisnummer nul werd ingegeven.

Op regel 90 wordt het record van de betreffende kennis ingelezen. Door deze GET-intstruktie worden de op regel 20 genoemde variabelen (NN$, SS$, WW$, en TT$) automatisch gevuld met de gegevens van deze kennis. Ook wordt MM$ gevuld, het merkteken dat aangeeft of er zinnige informatie in het record aanwezig is. Indien MM$ ongelijk is aan een sterretje, wordt dit veld alsnog gelijk gesteld aan een sterretje terwijl de andere velden uit het record worden schoongemaakt. Merk op dat de velden uit een record met een speciale instruktie (in dit geval LSET) dienen te worden gevuld. De velden uit een met behulp van het sleutelwoord FIELD gedefinieerd record mogen niet zomaar met een LET- of INPUT-kommando worden behandeld. Daarom worden deze velden vervolgens (110-140) ook overgenomen in de

velden N$, S$, W$ en T$.

Op regel 1/150-190 worden de ingelezen gegevens op beeldscherm afgedrukt. Vervolgens kunnen op regel 200-240 de gegevens opnieuw worden ingegeven. Doordat op de plaatsen van de INPUT-kommando's al gegevens staan vermeld, worden deze automtisch als waarde voor de INPUT genomen wanneer alleen een RETURN-toets wordt ingegeven. Ook kunnen deze gegevens met behulp van de diverse besturingstoetsen worden gecorrigeerd.

Op regel 250 en 260 worden de recordvelden met het LSET-kommando gelijk gemaakt aan de ingegeven waarden waarna op regel 270 het record in het kennissenbestand op de plaats wordt gezet die door de variabele K (kennisnummer) wordt aangegeven.

Uiteindelijk (GOTO 40) kan weer een nieuw kennisnummer worden ingegeven.

In tegenstelling tot het vorige voorbeeld kwamen we in dit laatste voorbeeld wat nieuwe sleutelwoorden tegen. De belangrijkste sleutelwoorden die we in verband met random files gaan tegenkomen zijn:

FIELD	voor het definiëren van de opbouw van een record
LSET	voor het invullen van een veld
RSET	ook voor het invullen van een veld, maar nu rechts aangeschoven
GET	voor het lezen van een record
PUT	voor het schrijven van een record

Voor een uitgebreide behandeling van deze sleutelwoorden dient hoofdstuk 4 te worden geraadpleegd.

3.13 De grootte van files

Het zal de oplettende lezer zijn opgevallen dat in de voorbeelden nergens op enigerlei wijze de grootte van een bestand werd aangegeven. MSX kent een zogenaamde DYNAMISCHE behandeling van bestanden. Hierdoor behoeft een grootte van een bestand niet te worden opgegeven; waar nodig vult MSX een bestand steeds aan tot de juiste grootte. De enige grens die er aan de grootte van een bestand ligt, is de opslagcapaciteit van de magneetschijf. Een eenvoudige rekensom leert dat er op een floppy disk van bijvoorbeeld 250 kilobytes een aantal

van ongeveer 2500 kennissen met hun namen, straten, woonplaatsen en telefoonnummers kunnen worden opgeslagen. De benadering van elke kennis afzonderlijk met behulp van het bovenstaande voorbeeldprogramma kost gemiddeld nog geen seconde...

4 DE MSX-DISK SLEUTELWOORDEN EN HUN BETEKENIS

In dit hoofdstuk worden de MSX-disk sleutelwoorden behandeld. Voorafgaand aan deze behandeling worden eerst wat sleutelwoorden behandeld die in het grote MSX-handboek reeds werden behandeld maar in MSX-disk basic een extra funktie krijgen. Daarna worden de nieuwe sleutelwoorden behandeld. Tenslotte worden nog enkele sleutelwoorden behandeld die door geen MSX computerfabrikant worden gedokumenteerd en veelal niet helemaal goed werken. Het lijkt erop dat deze sleutelwoorden in een latere versie van MSX-basic nog een funktie gaan krijgen...

De schrijfwijzen worden in de BNF-notatie gepresenteerd. Deze notatievorm werd in het grote MSX-handboek uitgebreid behandeld.

Omdat het hier slechts om een beperkt aantal sleutelwoorden gaat, worden deze, in tegenstelling tot in het grote MSX-handboek, hier in aanbevolen leervolgorde gepresenteerd. In hoofdstuk 5 vindt u desgewenst een alfabetische opsomming met de bijbehorende paginanummering.

Per nieuw sleutelwoord worden de volgende gegevens verstrekt:

- de naam van het sleutelwoord
- de moeilijkheidsgraad bij gebruik van dit sleutelwoord. Deze moeilijkheidsgraad varieert van zeer eenvoudig tot zeer moeilijk.
- de soort. We onderscheiden binnen de sleutelwoorden de A-FUNKTIES (funkties met een alfanumeriek resultaat), de N-FUNKTIES (funkties met een numeriek resultaat), SYSTEEMVARIABELEN (variabelen met een voorbestemde waarde) en KOMMANDO'S (sleutelwoorden die een aktie aangeven.
- de afkomst. Alle in MSX-basic voorkomende sleutelwoorden zijn afkomstig uit de Engelse taal. Het is vaak gemakkelijker om een sleutelwoord en de bijbehorende betekenis te onthouden wanneer men de preciese afkomst kent. Daarom is van elk sleutelwoord de afkomst naar het Nederlands herleid.
- de schrijfwijze in de BNF-notatievorm.

– de betekenis. De funktie van elk sleutelwoord wordt uitvoerig behandeld. Van sleutelwoorden die reeds in het grote MSX-handboek werden behandeld, is slechts een aanvullende beschrijving opgenomen.

– een voorbeeld. Waar zinvol is een voorbeeld opgenomen.

4.1 Enkele oude sleutelwoorden wat nader behandeld

OPEN

---

aanvullende betekenis

Het OPEN-kommando wordt gebruikt om een bestand, een groep van bij elkaar behorende gegevens, in te richten c.q. te kunnen gebruiken.

Het beeldscherm (CRT:), het grafische beeldscherm (GRP:), de printer (LPT:) of de cassetterecorder (CAS:) leerden we in het grote MSX-handboek reeds kennen. Als nieuwe randapparaten introduceert MSX-basic "A:...", "B:...", "C:..." enzovoorts als aanduidingen voor de eerste schijveneenheid, de tweede, de derde enzovoorts.

Als aanduiding op welke wijze het bestand dient te worden benaderd, zagen we reeds FOR OUTPUT en FOR INPUT. Als bijkomende aanduiding introduceert MSX-basic de aanduiding FOR APPEND waarmee we aangeven dat een bestand dient te worden AANGEVULD met gegevens.

de schrijfwijze dient er in BNF dus iets anders uit te zien:

```
                              INPUT ]
OPEN<BESTANDSNAAM>[FOR  OUTPUT ]AS[#]<KA-
   NAAL>                      APPEND]
```

Allereerst volgt hieronder een uitgebreide tabel met voorwaarden in verband met het OPEN-kommando:

randapparaat	betekenis	benadering	verplicht?	bestandsnaam verplicht?	kommentaar
CAS:	cassette-recorder	for input for output	ja	nee*	een cassetterecor-recorder moet zijn aangesloten
GRP:	grafisch scherm	for output	nee	nee, zinloos	een grafisch scherm moet zijn geaktiveerd
CRT:	alfanume-riek scherm	for output	nee	nee, zinloos	een alfanumeriek scherm moet zijn geaktiveerd
LPT:	printer	for output	nee	nee, zinloos	een printer moet zijn aangesloten
A:/B:/...	disk	for input for output for append	nee**	ja	een disk unit moet zijn aangesloten
(niets)	disk	. . . . zie A:/B:/...			

* indien geen bestandsnaam wordt opgegeven:
- bij FOR OUTPUT wordt een bestand zonder naam aangemaakt
- bij FOR INPUT wordt het eerste bestand dat op de band wordt tegengekomen, geopend

** indien geen benaderingswijze wordt gespecificeerd, wordt een RANDOM benaderingswijze aangenomen. Zie hiervoor de behandeling in hoofdstuk 3 en de sleutelwoorden vanaf FIELD.

Pas op dat een kommando als OPEN "FILE" AS 1, dat in MSX-basic een bestand op cassetteband opende, nu plotseling een bestand op schijf opent of creëert.

Voor de disk unit volgen hieronder enkele voorbeelden, gekombineerd met de behandeling van de overige 'oude' sleutelwoorden.

MAXFILES

Met de schijveneenheid krijgt de systeemvariabele MAXFILES pas echt een betekenis. Doordat de schijf random benaderbaar is, is het mogelijk om meerdere bestanden tegelijk op één schijf te creëren en/of te openen. Zelfs lees- en schrijfopdrachten kunnen gekombineerd worden.

Wanneer meerdere bestanden dienen te worden geopend, dan dient de systeemvariabele MAXFILES gelijk te worden gesteld aan het aantal tegelijk te openen bestanden. Een voorbeeld:

```
NEW
Ok
10 MAXFILES=2
20 OPEN "FILE1" FOR INPUT AS 1
30 OPEN "FILE2" FOR OUTPUT AS 2
40 ...
RUN
```

Op regel 10 werd bepaald dat er maximaal twee bestanden tegelijk mogen worden geopend. Op regel 20 werd vervolgens het bestand FILE1 geopend. Indien het bestand niet aanwezig is op de disk, volgt een foutmelding. Op regel 30 werd het bestand "FILE2" geopend en meteen schoongemaakt (met FOR OUTPUT geeft men aan dat men NIEUWE gegevens in het bestand wil schrijven). Indien bestand FILE2 niet aanwezig was, werd het aangemaakt.
Op regel 40 vervolgt het programma tenslotte.

CLOSE

Voordat we dit sleutelwoord verder behandelen eerst een waarschuwing:

Een bestand dat nog niet is gesloten, is meestal niet helemaal bijgewerkt. Indien een disk wordt vervangen door een andere terwijl bestanden nog niet gesloten zijn, kan het gebeuren dat de eerste schijf niet volledig wordt bijgewerkt terwijl de tweede wordt 'opgeblazen' (de gegevens worden vaak volledig verminkt). Nadat een programma is onderbroken, kunnen bestanden toch nog openstaan; pas bij een poging om het programma te veranderen, kan het plotseling gebeuren dat de disk unit weer even gaat werken; de openstaande bestanden worden dan als nog gesloten.

Alvorens schijven te verwisselen, moet u zich aanwennen om eerst een close-kommando voor de zekerheid in te toetsen zonder regelnummer.

Verwijder pas na de Ok-melding de schijf; er kan dan niets meer fout gaan.

Dus tik eerst in:

```
CLOSE
Ok
```

en verwijder dan pas de schijf.

Het CLOSE-kommando sluit een specifiek bestand; maakt het verder onbenaderbaar voor het programma. Alle kanalen kunnen in één keer worden gesloten door uitvoering van een enkele CLOSE. Door achter het sleutelwoord CLOSE een kanaalnummer te vermelden (eventueel voorafgegaan door een hekje) kan een specifiek bestand worden gesloten, namelijk het bestand dat eerder op het genoemde kanaalnummer werd geopend.

SAVE

Met het SAVE-kommando kan een programma op een randapparaat worden veiliggesteld.

Het SAVE-kommando krijgt in het MSX-disk basic een iets andere funktie dan in het MSX-basic:

MSX-basic	MSX-disk-basic
Indien in de programmanaam geen randapparaat wordt gespecificeerd, dan wordt de cassetterecorder als randapparaat aangenomen. De bevelen:	Indien in de programmanaam geen randapparaat wordt gekodeerd, dan wordt de laatste aktieve disk (meestal A:) als randapparaat genomen. De bevelen:
SAVE "PROG"	SAVE "PROG"
en	en
SAVE "CAS:PROG"	SAVE "A:PROG"

hebben dus dezelfde uitwerking. Het programma wordt altijd ongekodeerd, teken voor teken volgens de ASCII-kodering geschreven.	hebben dus dezelfde uitwerking (wanneer de schijf A: tenminste het laatst aktief was). Indien het programma naar disk wordt geschreven, dan gebeurt dat in de kodering waarmee het ook in het geheugen staat (verkort). Indien het programma NIET naar disk wordt geschreven, dan gebeurt dit via de ASCII-kodering.

Voor de bestandsnaam bij een SAVE-kommando gelden dezelfde eisen als genoemd bij het OPEN-kommando. Wanneer bij een SAVE naar schijf de ,A-optie wordt opgenomen, dan wordt het programma toch in ASCII-kodering geschreven. Dit is van belang wanneer het programma later met een MERGE bij een ander programma dient te worden gevoegd.

Dus:

```
SAVE "A:PROG",A
Ok
```

schrijft het programma PROG op schijf maar dan wel in ASCII-kodering; dezelfde kodering die met een SAVE op band wordt gebruikt.

Indien met behulp van SAVE een programma met de naam:

```
AUTOEXEC.BAS
```

op de schijf wordt geschreven, dan zal de computer mits de schijf met dit programma zich dan in de eerste schijveneenheid bevindt, de volgende keer dat hij wordt aangezet, onmiddellijk beginnen met het uitvoeren van dit programma.

LOAD

Het LOAD-kommando werkt als eerder in het grote MSX-handboek

beschreven met dit verschil dat wanneer geen randapparaat werd gespecificeerd, automatisch de laatst benaderde schijf wordt genomen. Zowel programma's die met de ,A-optie zijn geSAVEd als de programma's die zonder deze optie op schijf werden gezet, worden geladen.

RUN

Een met SAVE vastgelegd programma kan met een RUN "(programmanaam)" worden geladen en direkt worden opgestart. Onder MSX-diskbasic wordt indien geen randapparaat werd gespecificeerd, wederom de laatst benaderde schijf genomen. Zowel programma's mét als zónder ,A-optie kunnen op deze wijze worden geaktiveerd.

MERGE

Het MERGE-kommando werkt als eerder in het grote MSX-handboek beschreven met dit verschil dat wanneer geen randapparaat wordt gespecificeerd, automatisch de laatst benaderde schijf wordt genomen. Wanneer een programma met MERGE van schijf wordt geladen, dan dient dat programma eerder met de ,A-optie op deze schijf te zijn geSAVEd.

PRINT#

Het PRINT#-kommando heeft PRECIES dezelfde werking in het MSX-disk basic als in het gewone MSX-basic. Het OPEN-kommando bepaalt op welk randapparaat het PRINT#-kommando wordt uitgevoerd; het printkommando zelf verandert niet.

(LINE) INPUT#

Ook het (LINE) INPUT#-kommando behoudt in het MSX-disk basic precies dezelfde werking. Door het open-kommando wordt bepaald vanuit welk randapparaat gegevens worden ingelezen.

EOF(...)

De N-FUNKTIE EOF behoudt eveneens dezelfde funktie; het OPEN-kommando is bepalend voor het type randapparaat.

Tot slot volgen hier onder enkele kleine voorbeelden waarin op schijf wordt gewerkt en waarin de 'oude', hier boven behandelde sleutelwoorden worden gebruikt.

voorbeeld 1

In dit voorbeeld wordt bestand BES01 op de schijf toegewezen en schoongemaakt. Vervolgens kunnen regels tekst worden ingegeven die dan in dit bestand worden weggeschreven. Als zodanig kan bijvoorbeeld een brief op schijf worden bewaard.

```
NEW
Ok
10 REM VOORBEELD 1, INGAVE TEKST
20 CLS:OPEN "BES01" FOR OUTPUT AS 1
30 PRINT "GEEF TEKST IN (*=EINDE)"
40 LINE INPUT R$:IF R$="*" THEN 60
50 PRINT #1,R$:GOTO 40
60 CLOSE:STOP
RUN

(beeldscherm wordt schoongemaakt)

GEEF TEKST IN (*=EINDE)

Beste lezer,

Met deze brief testen we ons eerste
```

```
voorbeeld even uit. Hopelijk komt
deze tekst netjes in bestand BES01
te staan.

Hoogachtend,

Uw auteur.

*
Break in 60
Ok
```

voorbeeld 2

In dit voorbeeld wordt bestand BES01 op schijf geopend waarna regels tekst kunnen worden AANGEVULD in dit bestand.

```
NEW
Ok
10 REM VOORBEELD 2, AANVULLEN TEKST
20 CLS:OPEN "BES01" FOR APPEND AS 1
30 PRINT "GEEF TEKST IN (*=EINDE)"
40 LINE INPUT R$:IF R$="*" THEN 60
50 PRINT #1,R$:GOTO 40
60 CLOSE:STOP
RUN

(beeldscherm wordt schoongemaakt)

GEEF TEKST IN (*=EINDE)

P.S.: Als het goed is, wordt deze
      tekst nog aangevuld...

*
Break in 60
Ok
```

voorbeeld 3

In dit voorbeeld wordt bestand BES01 geopend en bestand BES02 toegewezen en schoongemaakt op schijf. Vervolgens worden de regels tekst die in BES01 liggen opgeslagen, stuk voor stuk 'overgeheveld' in bestand BES02. Er kan worden bepaald welke regels wel en niet worden overgeheveld. Ook kunnen regels worden tussengevoegd.

```
NEW
Ok
10 REM VOORBEELD 3, OVERHEVELEN
20 MAXFILES=2
30 OPEN "BES01" FOR INPUT AS 1
40 OPEN "BES02" FOR OUTPUT AS 2
50 CLS:PRINT "0=REGEL OVERSLAAN"
60 PRINT "1=REGEL OVERNEMEN"
70 PRINT "3=NA DEZE REGEL TUSSENVOEGEN"
80 PRINT
90 IF EOF(1) THEN 190
100 LINE INPUT #1,R$
110 PRINT R$
120 K$=INKEY$:IF K$="" THEN 120
130 IF K$<>"0" AND K$<>"1" AND K$<>"2" T
HEN BEEP:GOTO 120
140 IF K$="0" THEN PRINT "VERWIJDERD":BE
EP:GOTO 100
150 PRINT #2,R$:IF K$="1" THEN 90
160 REM TUSSENVOEGEN
170 PRINT"TUSSENVOEGEN (*=EINDE)"
180 LINE INPUT R$:IF R$="*" THEN PRINT "
EINDE TUSSENVOEGEN":GOTO 90 ELSE PRINT #
2,R$:GOTO 180
190 CLOSE:STOP
RUN

(beeld wordt schoongemaakt)
```

```
0=REGEL OVERSLAAN
1=REGEL OVERNEMEN
3=NA DEZE REGEL TUSSENVOEGEN


Beste lezer,

TUSSENVOEGEN (*=EINDE)
Allereerst mijn hartelijke dank voor
de aanschaf van dit boekwerkje.

*
EINDE TUSSENVOEGEN
Met deze brief testen we ons eerste
voorbeeld even uit. Hopelijk komt
deze tekst netjes in bestand BES01
te staan.

Hoogachtend,

Uw auteur.


P.S.: Als het goed is, wordt deze
TUSSENVOEGEN (*=EINDE)
        laatste regel verwijderd...
*
EINDE TUSSENVOEGEN
        tekst nog aangevuld...
VERWIJDERD

Break in 190
Ok
```

voorbeeld 4

In dit voorbeeld wordt BES02 in BES01 terug gekopieerd en vervolgens verwijderd. Zo kan het programma in voorbeeld 3 steeds worden herhaald.

```
NEW
Ok
10 REM VOORBEELD 4, BES02->BES01
20 MAXFILES=2
30 OPEN "BES02" FOR INPUT AS 1
40 OPEN "BES01" FOR OUTPUT AS 2
50 IF EOF(1) THEN 70
60 LINE INPUT #1,R$:PRINT #2,R$:GOTO 50
70 CLOSE:STOP
RUN
Break in 70
Ok
```

voorbeeld 5

In dit voorbeeld wordt bestand BES01 regel voor regel ingelezen en op beeldscherm afgedrukt. Door op regel 40 de PRINT te vervangen door een LPRINT kan worden bewerkstelligd dat de regels op een eventuele printer worden afgedrukt.

```
NEW
Ok
10 REM VOORBEELD 5, PRINT BES01
20 CLS:OPEN "BES01" FOR INPUT AS 1
30 IF EOF(1) THEN CLOSE:STOP
40 LINE INPUT#1,R$:PRINT R$:GOTO 30
RUN

(beeld wordt schoongemaakt)

Beste lezer,
```

```
Allereerst mijn hartelijke dank voor
de aanschaf van dit boekwerkje.

Met deze brief testen we ons eerste
voorbeeld even uit. Hopelijk komt
deze tekst netjes in bestand BES01
te staan.

Hoogachtend,

Uw auteur.


P.S.: Als het goed is, wordt deze
      laatste regel verwijderd...

Break in 30
Ok
```

Deze vijf programma's bij elkaar vormen een héél eenvoudig tekstverwerkingspakketje. Misschien een uitdaging voor de amateur om het een en ander verder uit te bouwen en te verfraaien...

4.2 De nieuwe sleutelwoorden

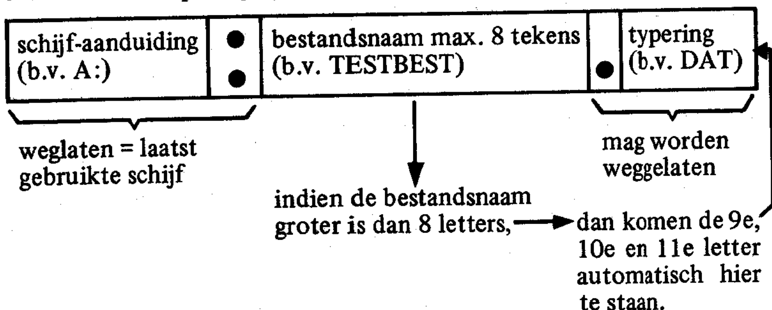
In deze paragraaf worden de sleutelwoorden behandeld die het MSX-disk basic extra biedt.
Echter, voordat we tot deze behandeling overgaan, dient eerst een veel voorkomende term nader te worden uitgelegd en wel de term:

⟨BESTANDSAANDUIDING⟩

Een bestandsaanduiding is een naamgeving waarmee we geen, één of een groep van bestanden kunnen aanduiden. Dit in tegenstelling tot een bestandsnaam, waarmee we precies één bestand kunnen aanduiden.

Om verder op deze term in te gaan, dienen we ons eerst te realiseren welke componenten een bestandsnaam op schijf in zich heeft of in zich kan hebben:

bestandsnaam: op schijf (b.v. A:TESTBEST.DAT)



Allereerst onderscheiden we in de bestandsnaam de schijfaanduiding (A:/B:/enz.).
Vervolgens onderscheiden we de naam van het bestand.
Achter de naam mag vervolgens een punt worden opgenomen, gevolgd door een drie-letterige typeaanduiding. Deze type-aanduiding is geheel vrij en mag naar believen worden toegepast. Het is echter raadzaam om bij het programmeren een vaste regel in verband met deze typering te hanteren. Bijvoorbeeld:

– een programma op schijf (geen ,A)	XXXXXXXX.BAS
– een programma op schijf (met ,A)	XXXXXXXX.ASC
– een bestand met daarin alleen tekst	XXXXXXXX.TXT
– een sequentieel bestand	XXXXXXXX.SEQ
– een random bestand	XXXXXXXX.RAN

Door alle bestanden van een bepaalde soort ook een bepaalde type-aanduiding te geven, zijn deze later (wanneer er tientallen bestanden op schijf staan) weer snel bij elkaar te zoeken.

Een programma kan dus bijvoorbeeld als volgt worden geSAVEd:

```
SAVE "A:VB01.BAS"
```

Met A: geven we aan dat het programma op de eerste schijf moet worden geschreven. Het programma zelf heet VB01 (voorbeeld 1) en is door middel van .BAS tot basic programma getypeerd.

```
SAVE ":OVL.ASC",A
```

Het programma OVL wordt op de tweede schijveneenheid geschreven. Omdat het een ASCII-bestand is (de ,A-optie), krijgt het programma de typering ASC mee.

Schijf-aanduiding en typering mogen worden weggelaten. Indien de schijfaanduiding wordt weggelaten, wordt de laatst aangesproken schijveneenheid aangenomen.

In MSX-disk basic is het bij sommige sleutelwoorden noodzakelijk om een groep van bestanden tegelijk te kunnen aanduiden. Dit doen we met een zogenaamde bestandsaanduiding.

Een bestandsaanduiding kan allereerst een schijf-aanduiding bevatten (bijvoorbeeld A: of B:) maar dit is niet noodzakelijk.

Vervolgens kan de bestandsaanduiding een bestandsnaam hebben. Deze mag geheel worden ingevuld, maar sommige tekens mogen worden vervangen door een vraagteken. De groep van bestanden die met die bestandsaanduiding wordt bedoeld, mogen in hun naam op deze positie elke letter bevatten.

Vervolgens kan een type-aanduiding worden opgenomen. Voor deze typering geldt eveneens dat sommige tekens door een vraagteken mogen worden vervangen. Ook mag een typering geheel worden weggelaten.

Enkele voorbeelden:

bestandsaanduiding	groep van bestanden die wordt aangeduid
"BES??"	alle bestanden op de laatst benaderde schijveneenheid die een naam van 5 letters hebben waarvan de eerste drie letters gelijk zijn aan BES
"A:?FILE"	alle bestanden op schijveneenheid A: (de eerste schijveneenheid) met een naam van 5 letters waarvan de laatste vier letters gelijk zijn aan FILE
"A:????.BAS"	alle bestanden op schijveneenheid A: met een naam van vier letters en een typering BAS

A:PROGRAMM.???	alle bestanden op schijveneenheid A: met de naam PROGRAMM en een typering van drie letters

Zowel de bestandsnaam als de typering mogen als laatste karakter een sterretje bevatten. Met dit sterretje geven we aan, dat de rest van de bestandsnaam of de typering er niet toe doet. Bijvoorbeeld:

bestandsaanduiding	groep van bestanden die wordt aangeduid
"BES*.DAT"	alle bestanden met een naam die begint met BES. De lengte van de naam doet er verder niet toe. Wel dient het bestand een typering DAT te hebben. De bestanden dienen aanwezig te zijn op de laatst benaderde schijveneenheid
"*.P*"	alle bestanden met een typering, beginnende met een P op de laatst aangesproken schijf
"A:*.*"	alle bestanden op schijveneenheid A:

Uiteraard mag een bestandsaanduiding ook gewoon een bestandsnaam bevatten. De 'groep' van bestanden die wordt bedoeld, bevat dan altijd maar één bestand.

SLEUTELWOORD **FILES**

moeilijkheidsgraad . . . . normaal
soort . . . . KOMMANDO
afkomst . . . . FILES is bestanden

schrijfwijze

```
FILES[<BESTANDSAANDUIDING>]
<BESTANDSAANDUIDING>::=<A>
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het sleutelwoord FILES kan worden bekeken, welke bestanden zich op een schijf bevinden. Voorbeelden:

`FILES`	laat zien welke bestanden zich op de laatst aangesproken schijf bevinden
`FILES "*.DAT"`	laat zien welke bestanden zich op de laatst aangesproken schijveneenheid bevinden met een typering DAT
`FILES "A:P*.*"`	laat alle bestanden op schijveneenheid A: zien waarvan de namen met een P beginnen
`FILES "A:TEST.BAS"`	laat zien of bestand TEST.BAS op de eerste schijveneenheid voorkomt

Het volgende voorbeeld laat de inhoud zien van zo maar een schijf:

```
FILES
EDIT            V07
V02             V01
V03             FRAME
EKEN            TEKEN
TEK       .DAT  TEKRES
OVL             V04
TEKERS          V05
```

```
V06                 TEL        .DAT
V08                 V011
V09                 V010
Ok
```

SLEUTELWOORD LFILES

moeilijkheidsgraad .... **normaal**
soort .... **KOMMANDO**
afkomst .... **LFILES is samentrekking van line printer en files – bestanden (bestandsnamen) op afdrukeenheid (printer) afdrukken**

schrijfwijze

```
LFILES[<BESTANDSAANDUIDING>]
<BESTANDSAANDUIDING>::=<A>
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord heeft op één aspekt na precies dezelfde werking als het sleutelwoord FILES. Het verschil is, dat LFILES het bedoelde overzicht op een eventueel aangesloten printer afdrukt in plaats van op het beeldscherm.

SLEUTELWOORD KILL

moeilijkheidsgraad .... **normaal**
soort .... **KOMMANDO**
afkomst .... **KILL is doden, elimineren**

schrijfwijze

```
KILL<BESTANDSAANDUIDING>
<BESTANDSAANDUIDING>::=<A>
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het sleutelwoord KILL kunnen we een bestand of een groep van bestanden volledig en in één klap van schijf verwijderen. Bijvoorbeeld:

`KILL "BEST.DAT"`	verwijdert bestand BEST.DAT van de schijf
`KILL "*.DAT"`	verwijdert alle bestanden van schijf met typering DAT
`KILL "A:*.*"`	verwijdert alle bestanden van schijveneenheid A:

Het spreekt vanzelf dat met het sleutelwoord KILL met grote voorzichtigheid dient te worden omgesprongen.

SLEUTELWOORD # COPY

moeilijkheidsgraad . . . . **normaal**
soort . . . . **KOMMANDO**
afkomst . . . . **COPY is kopieren**

schrijfwijze

```
COPY<BESTANDSAANDUIDING>[TO<BESTANDS-
  AANDUIDING>]
<BESTANDSAANDUIDING>::=<A>
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het sleutelwoord COPY kunnen complete bestanden of groepen van bestanden worden gekopiëerd. Bijvoorbeeld:

```
COPY "A:BES1" TO A:BES2"
```

Bestand BES1 wordt gekopiëerd onder de naam BES2 zodat nu twee identieke bestanden onder twee verschillende namen op de eerste schijf zijn opgeslagen.

```
COPY "T*" TO "Q*"
```

Alle bestanden waarvan de naam begint met een T en die geen typering hebben, worden op dezelfde schijf gekopiëerd onder een andere naam. De nieuwe naam is gelijk aan de oude naam op één letter na. De eerste letter, oorspronkelijk een T, is nu namelijk door een Q vervangen.

```
COPY "A:*.BAS" TO "A:*.PRG"
```

Alle bestanden op schijveneenheid A: met typering BAS worden op dezelfde schijf onder dezelfde namen maar onder een andere typering (PRG) gekopiëerd.

```
COPY "B:" TO "A:"
```

Kopiëert alle bestanden van de tweede schijveneenheid naar de eerste schijveneenheid. Indien er slechts één schijveneenheid aanwezig is, zal MSX deze schijveneenheid steeds afwisselend de rol van eerste en tweede schijveneenheid laten spelen. Tussendoor vraagt MSX u auto-

matisch steeds om de schijf te verwisselen.

```
COPY "B:"
```

Indien TO... wordt weggelaten, neemt de MSX-computer automatisch aan dat er naar de laatst benaderde schijveneenheid dient te worden gekopieerd. Is dit schijveneenheid A:, dan zal de uitwerking van dit kommando dezelfde zijn als hierboven beschreven.

De COPY "B:" TO "A:" biedt ons de mogelijkheid om in één keer de hele schijf te kopiëren. Het volgende voorbeeld laat zien hoe dat in zijn werk gaat:

```
COPY "A:" TO "B:"

Insert diskette for drive B:
and strike a key when ready

Insert diskette for drive A:
and strike a key when ready
Insert diskette for drive B:
and strike a key when ready

Insert diskette for drive A:
and strike a key when ready
Ok
```

Steeds dient afwisselend de originele en de kopie-schijf te worden aangeboden aan de disk drive. Bij aanvang dient de originele schijf in de eenheid te zitten. Stel eventueel met WRITE PROTECT optie het origineel veilig.

SLEUTELWOORD # NAME

moeilijkheidsgraad . . . . **eenvoudig**
soort . . . . **KOMMANDO**
afkomst . . . . **NAME is naam geven**

schrijfwijze

```
NAME<BESTANDSAANDUIDING>AS<BESTANDS-
  AANDUIDING>
<BESTANDSAANDUIDING>::=<A>
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het sleutelwoord NAME kan een bestand of een groep van bestanden van een nieuwe naam worden voorzien. Bijvoorbeeld:

```
NAME "A:FILE.BEST" AS "BEST.FIL"
```

verandert de naam van FILE.BES in BEST.FIL op schijf A:

```
NAME "A:*.BAS" AS "*.PRG"
```

verandert de typeringen van de betreffende bestanden van BAS naar PRG.

```
NAME "B???" AS "Q???"
```

op de laatst aangesproken schijveneenheid worden de bestanden met een naam van vier letters beginnende met een B van een andere naam voorzien. De nieuwe naam van die bestanden begint met een Q en is verder gelijk aan de oude naam.

SLEUTELWOORD # DSKF

moeilijkheidsgraad .... **normaal**
soort .... **N-FUNKTIE**
afkomst .... **DKSF is samentrekking van disk en free – vrije ruimte op schijf**

schrijfwijze

```
DSKF(<SCHIJFNUMMER>)
<SCHIJFNUMMER>::=<A>
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het sleutelwoord DSKF kan de vrije ruimte die nog op een schijf over is, worden opgevraagd. Als schijfnummer kan het werkelijke schijfnummer worden opgegeven (1,2,3 ... komt overeen met A:,B:,C: ...) of kan een nulwaarde worden opgegeven, waarmee dan de laatst benaderde schijf kan worden bedoeld. Indien de numerieke uitdrukking die de schijveneenheid bepaalt, een decimale fraktie bevat, dan wordt deze verwaarloosd.

De waarde die de DSKF-funktie als resultaat geeft, kan verschillende betekenissen hebben, afhankelijk van het merk en type van de schijveneenheid. Meestal wordt een numerieke waarde toegekend die gelijk is aan het aantal KILOBYTES dat nog vrij is op de schijf. Bijvoorbeeld:

```
PRINT DSKF(0)
 273
Ok
SAVE "TEST"
Ok
PRINT DSKF(0)
 272
Ok
```

In het bovenstaande voorbeeld bleken op een bepaalde schijf nog 273 kilobytes aan ruimte over te zijn. Het daarna op schijf gezette programma bleek één kilobyte aan vrije ruimte te bezetten.

SLEUTELWOORD # LOF

moeilijkheidsgraad . . . . **normaal**
soort . . . . **N-FUNKTIE**
afkomst . . . . **LOF is afkorting van lengthe of file – lengte van het bestand**
schrijfwijze

```
LOF(<KANAAL>)
<KANAAL>::=<N>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het sleutelwoord LOF kan de lengte van een bestand worden bepaald. Tussen haakjes dient een uitdrukking te worden genomen die het kanaalnummer aangeeft waarop het betreffende bestand eerder werd geopend. Indien deze numerieke uitdrukking een decimale fraktie bevat, wordt deze verwaarloosd. Het resultaat van deze funktie is een waarde die de lengte van het gehele bestand in BYTES uitdrukt.

Bijvoorbeeld:

```
NEW
Ok
10 OPEN "TEK.DAT" AS 1
20 PRINT LOF(1)
30 CLOSE
RUN
 1876
Ok
```

In het bovenstaande voorbeeld bleek bestand TEK.DAT 1876 bytes lang te zijn.

SLEUTELWOORD **FIELD**

---

moeilijkheidsgraad . . . . **normaal**
soort . . . . **KOMMANDO**
afkomst . . . . **FIELD is veld, gebied**

schrijfwijze

```
FIELD[#]<KANAAL><RECORDINDELING>
<RECORDINDELING>::={,<VELDLENGTE>AS
  <VELDNAAM>}\<NIETS>
<KANAAL>::=<N>
<VELDLENGTE>::=<N>\<...><VARIABELE-
  NAAM>
<VELDNAAM>::=<ALFANUMERIEKE VARIABELE>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
<VARIABELE-NAAM>::=<ZIE ALGEMENE SPE-
  CIFICATIES>
<ALFANUMERIEKE VARIABELE>::=<ZIE AL-
  GEMENE SPECIFICATIES>
```

Wanneer we in MSX-basic met RANDOM FILES willen werken, dienen RECORDS te definiëren (zie de in hoofdstuk 3 behandelde stof). Het definiëren van records dienen we in MSX-basic met het FIELD-sleutelwoord te doen. Voorafgaand aan het FIELD-kommando dient een OPEN-kommando te zijn opgenomen dat een RANDOM-bestand op het betreffende kanaal opent. Dit openen dient *niet* met één van de toevoegingen FOR INPUT, FOR OUTPUT of FOR APPEND te geschieden.

Na het openen dient de recordsamenstelling met het FIELD-kommando te worden gedefiniëerd.

Na het FIELD-sleutelwoord dient eerst, al dan niet voorzien van een hekje, het kanaalnummer te worden opgenomen. Dit kanaalnummer moet in waarde gelijk zijn aan het kanaalnummer waarover het betreffende bestand werd geopend. Een eventuele decimale fraktie wordt verwaarloosd.

Vervolgens dienen per veld de lengten van dat veld (in bytes) en de naam van dat veld (gewoon de naam van een alfanumerieke variabele) te worden opgenomen.

Bijvoorbeeld:

```
NEW
Ok
10 OPEN "KENNIS" AS #1
20 FIELD #1,1 AS MM$,24 AS NN$,24 AS SS$
,24 AS WW$,24 AS TT$
```

In dit voorbeeld wordt op regel 10 eerst het bestand KENNIS geopend. Indien het betreffende bestand nog niet aanwezig was, wordt het gecreëerd.

Op regel 20 wordt van het RANDOM bestand KENNIS de recordsamenstelling bepaald. Het record bestaat uit een veld met MM$ (1 byte lang), een veld NN$ (24 bytes lang), een veld SS$ (24 bytes lang), een veld WW$ (24 bytes lang) en een veld TT$ (24 bytes lang). Het totale record is dus 1+24+24+24+24=97 bytes lang.

Het bovenstaande voorbeeld is een gedeelte van het voorbeeld dat in hoofdstuk 3 werd getoond. In de verdere behandeling zullen we ons waar mogelijk aan dit voorbeeld houden.

SLEUTELWOORD # LSET

moeilijkheidsgraad .... **normaal**
soort .... **KOMMANDO**
afkomst .... **LSET is afkorting van left set – links klaarzetten**

schrijfwijze

```
LSET<VELDNAAM>=<A>
<VELDNAAM>::=<ALFANUMERIEKE VARIABELE>
<ALFANUMERIEKE VARIABELE>::=<ZIE ALGE-
  MENE SPECIFICATIES>
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Variabelen die met behulp van het FIELD-kommando als naam zijn toegekend aan een veld uit een record, mogen niet zo maar op de gebruikelijke wijze aan een waarde worden gelijkgesteld. De FIELD-variabelen zijn zeer speciale variabelen in MSX-basic die alleen met het LSET, RSET of GET-kommando van waarde mogen worden veranderd. Wanneer we bijvoorbeeld zouden programmeren:

```
NEW
Ok
10 OPEN "KENNIS" AS #1
20 FIELD #1,1 AS MM$,24 AS NN$,24 AS SS$
,24 AS WW$,24 AS TT$
30 NN$="GERT-JAN JANSSEN"
```

dan koppelen we het veld NN$ af van het FIELD-kommando en wordt het verder onbruikbaar voor de bestandsbenadering.

Met het LSET-kommando kunnen we een waarde aan een recordveld toekennen. Deze waarde wordt dan LINKS in het recordveld aangesloten terwijl het recordveld vervolgens eventueel met spaties wordt aangevuld. Bijvoorbeeld:

```
250 LSET NN$=N$:LSET SS$=S$
260 LSET WW$=W$:LSET TT$=T$
```

Uitgaande van het voorbeeld, gegeven onder FIELD, worden in dit

voorbeeld de velden NN$, SS$, WW$ en TT$ gelijkgesteld aan N$, S$, W$ en T$. Alle vier de veldwaarden worden tot 24 posities aangevuld met spaties.

SLEUTELWOORD **RSET**

moeilijkheidsgraad .... normaal
soort .... KOMMANDO
afkomst .... RSET is afkorting van right set – rechts klaarzetten

schrijfwijze

```
RSET<VELDNAAM>=<A>
<VELDNAAM>::=<ALFANUMERIEKE VARIABELE>
<ALFANUMERIEKE VARIABELE>::=<ZIE ALGE-
   MENE SPECIFICATIES>
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Het RSET-sleutelwoord heeft ongeveer dezelfde betekenis als het LSET-sleutelwoord. Het verschil is, dat RSET de betreffende waarde RECHTS in het recordveld aansluit. Bijvoorbeeld:

```
NEW
Ok
10 OPEN "FILE" AS 1
20 FIELD #1,10 AS SA$
30 RSET SA$="123.45"
40 PRINT SA$
50 STOP
RUN
    123.45
Break in 50
Ok
```

Bovenstaand voorbeeld laat zien dat een met behulp van RSET aan een veld toegekende waarde rechts uitgelijnd wordt geplaatst.

SLEUTELWOORD PUT

moeilijkheidsgraad . . . . normaal
soort . . . . KOMMANDO
afkomst . . . . PUT is zetten, plaatsen

schrijfwijze

```
PUT[#]<KANAAL>[,<RECORD>]
<KANAAL>::=<N>
<RECORD>::=<N>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met het PUT-kommando kan een kompleet record (dus meerdere velden) in het bestand worden geschreven. Het record dat wordt geschreven, moet in een FIELD-kommando zijn gespecificeerd terwijl alle velden met een LSET of een RSET (of eventueel een GET) dienen te zijn gevuld.

Achter het PUT-kommando dient eerst eventueel een hekje en daarna het kanaalnummer te worden opgenomen. Een eventuele decimale fraktie wordt verwaarloosd. Vervolgens kan een recordnummer worden opgenomen. Dit recordnummer bepaald als hoeveelste record het betreffende record in het bestand wordt geschreven. Een eventuele decimale fraktie wordt verwaarloosd.

Uitgaande van het voorbeeld, gegeven onder FIELD en LSET, schrijft het kommando:

```
270 PUT #1,R
```

het record, bestaande uit MM$, NN$, SS$, WW$ en TT$, in bestand KENNIS. Als hoeveelste record het betreffende record wordt geschreven, wordt bepaald door de variabele R.

Wanneer een recordnummer wordt weggelaten, dan wordt het record automatisch als volgende record geschreven. Indien na het openen van het bestand nog niet eerder een PUT of GET werd uitgevoerd, dan wordt het eerste record geschreven.

SLEUTELWOORD GET

moeilijkheidsgraad .... normaal
soort .... KOMMANDO
afkomst .... GET is krijgen, pakken

schrijfwijze

```
GET[#]<KANAAL>[,<RECORD>]
<KANAAL>::=<N>
<RECORD>::=<N>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Het GET-kommando werkt precies tegenovergesteld aan het PUT-kommando; de velden van een bepaald record worden juist ingelezen. Uitgaande van het voorbeeld, gegeven onder FIELD, LSET en PUT, leest het kommando:

```
280 GET #1,R
```

het record weer in. Door dit ene, eenvoudige kommando worden de velden MM$, NN$, SS$, WW$ en TT$ weer vanaf de schijf ingelezen.

Indien het recordnummer wordt weggelaten, wordt het volgende (of eerste) record ingelezen (als PUT).

Met de tot nu toe behandelde kommando's en funkties kan al volledig worden gewerkt met random bestanden.

In hoofdstuk 3 werd reeds een voorbeeld van een bestandsonderhouds-programma gegeven. Dat voorbeeld wordt hier nog eens geplaatst:

```
NEW
Ok
10 OPEN "KENNIS" AS #1
20 FIELD #1,1 AS MM$,24 AS NN$,24 AS SS$
,24 AS WW$,24 AS TT$
30 PUT #1,101
40 CLS:PRINT "VULLEN KENNISSENBESTAND"
```

```
50 LOCATE 0,3:INPUT "KENNISNUMMER";K
60 IF K=0 THEN CLOSE:STOP
70 IF K>100 THEN BEEP:GOTO 40
80 IF K<>INT(K) THEN BEEP:GOTO 40
90 GET #1,K
100 IF MM$<>"*" THEN LSET MM$="*":LSET N
N$=" ":LSET SS$=" ":LSET WW$=" ":LSET TT
$=" "
110 N$=NN$
120 S$=SS$
130 W$=WW$
140 T$=TT$
150 LOCATE 0,5
160 PRINT "NAAM? ";N$
170 PRINT "STRAAT? ";S$
180 PRINT "WOONPLAATS? ";W$
190 PRINT "TELEFOONNUMMER? ";T$
200 LOCATE 0,5
210 INPUT "NAAM";N$
220 INPUT "STRAAT";S$
230 INPUT "WOONPLAATS";W$
240 INPUT "TELEFOONNUMMER";T$
250 LSET NN$=N$:LSET SS$=S$
260 LSET WW$=W$:LSET TT$=T$
270 PUT #1,K
280 GOTO 40
RUN


(beeld wordt schoongemaakt)



VULLEN KENNISSENBESTAND
```

```
KENNISNUMMER? 3

NAAM? PIET JANSSEN
STRAAT? STRAATWEG 11
WOONPLAATS? SCHIEDAM
TELEFOONNUMMER? 010-125684
```

Op regel 10 wordt het bestand geopend en eventueel aangemaakt. Het bestand krijgt kanaal 1 toegewezen.

Op regel 20 wordt de recordindeling gespecificeerd. Het record bestaat in dit geval uit:

- MM$ dit veld bevat een sterretje indien het record door het programma eerder werd gevuld.
- NN$ dit veld bevat de naam van de kennis waarvan we de gegevens willen opslaan.
- SS$ dit veld bevat straat en huisnummer van de betreffende kennis.
- WW$ dit veld bevat postkode en woonplaats van de betreffende kennis.
- TT$ dit veld bevat het telefoonnummer van de betreffende kennis.

Veld MM$ kan maximaal 1 positie lang zijn; de overige velden hebben allemaal een maximale lengte van 24 posities.

Op regel 30 wordt record nummer 101 geschreven. In het bestand KENNIS willen we maximaal 100 kennissen opslaan. Om er voor te zorgen dat we later bij een GET van een nog niet eerder geschreven record een foutmelding krijgen, passen we de truc toe dat we het hoogst mogelijke record plus 1 schrijven. Alle onderliggende records worden, alhoewel ze met onzinnige gegevens gevuld kunnen zijn, dan wel als gedefiniëerd beschouwd. Een GET op één van deze records geeft dan geen foutmelding meer.

Op regel 40 wordt het beeldscherm opgemaakt en op regel 50 kan een kennisnummer worden ingegeven. Op regel 60, 70 en 80 wordt gekontroleerd of het ingegeven nummer wel groter dan 0, kleiner dan 101 en geheel is. Indien een nul werd ingegeven, wordt het programma na een CLOSE (afsluiten van het bestand) beëindigd.

Op regel 90 wordt het record met het ingegeven (kennis)nummer gelezen. Op regel 100 wordt vervolgens gekontroleerd of het record reeds eerder werd geschreven (MM$ moet dan gelijk zijn aan een *). Indien dit niet het geval is, worden de recordvelden allemaal met behulp van het LSET-kommando op spaties gesteld terwijl MM$ nu wel aan een sterretje wordt gelijkgesteld.

Op regel 110, 120, 130 en 140 worden de variabelen N$, S$, W$ en T$ gelijk gesteld aan NN$, SS$, WW$ en TT$. De recordvelden worden ter verdere bewerking overgeheveld in niet aan het record gekoppelde variabelen.

De regels 150, 160, 170, 180 en 190 dragen er zorg voor, dat de inhoud van het zojuist ingelezen record netjes op het beeldscherm wordt geprojekteerd. De regels 200, 210, 220, 230 en 240 staan vervolgens nieuwe ingaven voor de betreffende variabelen (N$, S$, W$ en T$) toe. Doordat deze ingaven (zie de LOCATE op 150 en 200) over de projektie heen wordt uitgevoerd, wordt bewerkstelligd dat de ingave van alleen een RETURN de betreffende variabele gelijkstelt aan de oude waarde. Ook kunnen de reeds geprojekteerde gegevens met de MSX full-screen editor worden bewerkt.

Dit over elkaar projekteren en ingeven is een leuke truc die in MSX mogelijk is en krachtig programmeren toestaat.

Op regel 250 en 260 worden op gebruikelijke wijze de recordvelden weer op de juiste waarden gebracht. De records worden netjes rechts aangevuld met spaties tot de geldende veldlengten.

Op regel 270 wordt het eerder opgehaalde record weer op dezelfde plaats in het bestand teruggeschreven waarna (GOTO 40) een volgend kennisnummer kan worden ingegeven.

Het spreekt voor zich dat dit programma naar eigen wens kan worden aangepast. De ervaren amateur maakt met behulp van dit programma in een kwartiertje zijn eigen bestandsonderhoudsprogramma waarin hij of zij de gegevens van de leden van een voetbalclub, een postzegelverzameling en dergelijke bijhoudt.

Het is echter raadzaam om deze pogingen nog niet al te serieus te ondernemen; achter in dit boekje is een programma opgenomen waarmee iedereen binnen enkele MINUTEN zijn eigen bestandsonderhoudsprogramma samenstelt...

SLEUTELWOORD LOC

moeilijkheidsgraad .... normaal
soort .... N-FUNKTIE
afkomst .... LOC is afkorting van location – locatie, plaats

schrijfwijze

```
LOC(<KANAAL>)
<KANAAL>::=<N>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Met de LOC-funktie kan worden opgevraagd, op welke positie in het betreffende bestand we ons op een bepaald moment bevinden.

Hiertoe dient tussen haakjes een uitdrukking te worden opgenomen die in waarde het kanaalnummer voorstelt waarover het te onderzoeken bestand eerder werd geopend. Een decimale fraktie wordt verwaarloosd.

De LOC-funktie kan, afhankelijk van het betreffende bestandstype, twee verschillende soorten waarden als resultaat geven:

een niet random bestand (geopend met FOR INPUT/OUTPUT/ APPEND)

LOC geeft het aantal bytes aan dat het bestand op het moment van opvragen groot is. LOC geeft altijd een veelvoud van 256 bytes waardoor LOC slechts een globale indikatie geeft van de grootte van het bestand.

Bijvoorbeeld:

```
NEW
Ok
10 OPEN "TEST" FOR OUTPUT AS 1
20 PRINT USING "### ";LOC(1);
30 PRINT #1,"DIT IS EEN TEST":GOTO 20
RUN
  0   0   0   0   0   0   0   0   0   0
```

```
  0    0    0    0    0    0 256 256 256 256
256 256 256 256 256 256 256 256 256 256
256 512 512 (...etcetera)
```

een random bestand (geopend zonder FOR INPUT/OUTPUT/APPEND)

LOC geeft het recordnummer aan van het laatst benaderde record. Bijvoorbeeld:

```
NEW
Ok
10 OPEN "KENNIS" AS 1
20 PRINT USING "### ";LOC(1);
30 GET 1:GOTO 20
RUN
  0   1   2   3   4   5   6   7   8   9
 10  11  12  13  14  15  16  17  18  19
 20  21  22  23  24  25  26  27  28  29
 30  31  32 (...etcetera)
```

Merk op dat een GET kan worden gegeven (30) zonder dat een record met FIELD is gedefiniëerd. Het binnengehaalde record is dan echter niet benaderbaar.

SLEUTELWOORD # MKI$

moeilijkheidsgraad . . .**vrij moeilijk, kennis van de opslagmethodiek van numerieke en alfanumerieke variabelen is een voordeel**
soort . . . . **A-FUNKTIE**
afkomst . . . .**MKI$ is afkorting van make integer value string – maak string van integere waarde**

schrijfwijze

```
MKI$(<N>)
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Wanneer we numerieke gegevens in een bestand willen schrijven, dienen we deze gegevens eerst te converteren naar een alfanumeriek gegeven. Pas nadat we het kommando:

```
LET V$=STR$(V)
```

hebben uitgevoerd, kunnen we de waarde van de variabele V in een record opnemen en vervolgens schrijven. Andersom wordt V na het lezen van dit record weer op de juiste waarde gezet door een:

```
LET V=VAL(V$)
```

Wanneer in bovenstaand voorbeeld de variabele V een waarde heeft van 12345678, dan zijn er minimaal acht bytes nodig om deze waarde volgens bovenstaand recept in een bestand te schrijven. Echter, in het computergeheugen beslaat deze variabele V maar 4 bytes aan geheugenruimte. Dit komt doordat de interne opslag van numerieke variabelen in het MSX-computergeheugen op een bepaalde wijze zijn gekodeerd, afwijkend van de ASCII-kodering.

Om nu dit verlies van ongeveer 100 % aan schijfruimte te kunnen voorkomen, biedt MSX de mogelijkheid om een variabele in een string te plaatsen zoals deze ook in het geheugen is gekodeerd. Hierdoor kan een integere waarde altijd in twee bytes, een waarde van enkelvoudige precisie in vier bytes en een waarde van dubbele precisie in acht bytes worden gekodeerd. Een bijkomend verschijnsel is, dat de conversie van numerieke waarde naar string-waarde door de computer op deze wijze

sneller kan worden uitgevoerd.

Het kommando:

```
LET V$=MKI$(V)
```

resulteert altijd in een alfanumerieke variabele van twee bytes lang. Wanneer we met een:

```
PRINT V$
```

de waarde van V$ proberen te achterhalen, krijgen we op het scherm slechts zinloze gegevens; de informatie is niet volgens de ASCII-kodering in de string opgenomen en dus niet zondermeer dekodeerbaar.

Echter, wanneer we intikken:

```
LET V=CVI(V$)
PRINT V
```

dan zal de computer met de juiste waarde reageren; de funktie CVI is in staat om van de schijnbaar zinloze informatie in V$ weer een juiste numerieke waarde te maken.

Voor deze optimale conversie van numerieke waarden naar strings en omgedraaid, kennen we zes verschillende funkties:

FUNKTIE	resultaat	lengte van de string	tegenovergestelde funktie
V$=MKI$(V)	een integere waarde wordt naar een alfanumerieke waarde geconverteerd	2 bytes	V=CVI(V$)
V$=MKS(V)	idem voor een waarde van enkelvoudige precisie	4 bytes	V=CVS(V$)
V$=MKD$(V)	idem voor een waarde van dubbele precisie	8 bytes	V= CVD(V$)

Steeds zijn als voorbeeld de variabelen V en V$ gebruikt; elke andere variabele is natuurlijk toegestaan.

Numerieke waarden kunnen in een random bestand als volgt worden verwerkt:

```
NEW
Ok
10 OPEN "TEST" AS 1
20 FIELD #1, 2 AS S$,4 AS T$,8 AS U$
30 S%=32767:S!=123456!:S#=1.234E+36
50 LSET S$=MKI$(S%):LSET T$=MKS$(S!)
60 LSET U$=MKD$(S#)
70 PUT 1,20:S%=0:S!=0:S#=0
80 GET 1,20:S%=CVI(S$):S!=CVS(T$)
90 S#=CVD(U$):PRINT S%;S!;S#
100 CLOSE:KILL "TEST":STOP
RUN
 32767  123456  1.234E+36
Break in 100
Ok
```

In het bovenstaande voorbeeld worden drie verschillende typen numerieke variabelen gevuld en geconverteerd naar string-variabelen. Deze stringvariabelen worden in bestand TEST geschreven. Vervolgens worden ze weer ingelezen en worden deze variabelen weer naar numerieke variabelen geconverteerd en afgedrukt. Uiteindelijk wordt bestand TEST van schijf verwijderd.

PAS OP:
Wanneer via MKI$%, MKS$ en MKD$ geconverteerde variabelen in een bestand dienen te worden geschreven, dan mag dat slechts een random bestand zijn. Het printen van dergelijke stringwaarden in een sequentieel bestand geeft onherroepelijk moeilijkheden bij het latere teruglezen.

SLEUTELWOORD # MKS$

moeilijkheidsgraad . . **vrij moeilijk, kennis van de opslagmethodiek van numerieke en alfanumerieke variabelen is een voordeel**
soort . . . . **A-FUNKTIE**
afkomst . . . . **MKS$ is afkorting van make single precision value string – maak string van enkelvoudige precisie variabele**

schrijfwijze

```
MKS$(<N>)
<N>::<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord wordt onder MKI$ behandeld; zie aldaar.

SLEUTELWOORD MKD$

moeilijkheidsgraad . . . vrij moeilijk, kennis van de opslagmethodiek van numerieke en alfanumerieke variabelen is een voordeel.

soort . . . . A-FUNKTIE

afkomst . . . . MKD$ is afkorting van make double precision value string – maak string van dubbele precisie variabele

schrijfwijze

```
MKD$(<N>)
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord wordt onder MKI$ behandeld; zie aldaar.

SLEUTELWOORD CVI

moeilijkheidsgraad . . vrij moeilijk, kennis van de opslag methodiek van numerieke en alfanumerieke variabelen is een voordeel

soort . . . . N–FUNKTIE

afkomst . . . . CVI is afkorting van convert to integer value – zet over naar integere waarde

schrijfwijze

```
CVI(<A>)
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord wordt onder MKI$ behandeld; zie aldaar.

SLEUTELWOORD # CVS

moeilijkheidsgraad . . **vrij moeilijk, kennis van de opslag methodiek van numerieke en alfanumerieke variabelen is een voordeel**

soort . . . . **N-FUNKTIE**

afkomst . . . . **CVS is afkorting van convert to single precision value – zet over naar enkelvoudige precisie waarde**

schrijfwijze

```
CVS(<A>)
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord wordt onder MKI$ behandeld; zie aldaar.

SLEUTELWOORD # CVD

moeilijkheidsgraad . . .vrij moeilijk, kennis van de opslagmethodiek van numerieke en alfanumerieke variabelen is een voordeel
soort . . . . N-FUNKTIE
afkomst . . . . CVD is afkorting van convert to double precision value – zet over naar dubbele precisie waarde

schrijfwijze

```
CVD(<A>)
<A>::=<ZIE ALGEMENE SPECIFICATIES>
```

Dit sleutelwoord wordt onder MKI$ behandeld; zie aldaar.

4.3 Wat minder belangrijke, oude sleutelwoorden wat nader behandeld

In de vorige paragrafen werden de meest belangrijke sleutelwoorden behandeld. Enkele wat minder belangrijke sleutelwoorden die in het grote MSX-handboek reeds werden behandeld, worden hier wat nader uitgewerkt:

VARPTR

Ook voor op schijf opgeslagen bestanden geeft deze funktie de waarde van het adres van het zogenaamde FILE CONTROL BLOCK in het geheugen.

CALL

Twee CALL's zijn met name belangrijk bij het gebruik van schijveneenheden:

1. Om schijven te kunnen formatteren (zie hoofdstuk 3) dient een CALL te zijn voorgeschreven door de fabrikant van de schijveneenheid. Meestal luidt deze call: CALL FORMAT.

2. Wanneer het MSX-basic vanuit een van schijf geladen disk operating system werd opgestart (MSX-DOS of CP/M) dan kan meestal door ingave van het kommando CALL SYSTEM naar dit operating system worden teruggekeerd.

BLOAD en BSAVE

Ook op schijf kunnen gegevens rechtstreeks uit het computergeheugen worden opgeslagen. BLOAD en BSAVE kiezen in de disk-versie van MSX automatisch voor de schijf indien geen ander randapparaat (b.v. CAS:) werd aangegeven.

4.4 Enkele merkwaardigheden

Het MSX-basic bevat in de 1.0 versie enkele verschijnselen die merkwaardig aandoen en waarschijnlijk fout zijn. Meestal gaat het om dingen die het noemen nauwelijks waard zijn.

Daarop is echter één uitzondering:

EEN RECORD IN EEN RANDOM FILE BESLAAT ALTIJD 256 BYTES, HOE DE INDELING DOOR HET FIELD-KOMMANDO OOK WERD BEPAALD.

Wanneer we bijvoorbeeld het volgende programma schrijven:

```
NEW
Ok
10 OPEN "TEST" AS 1
20 FIELD #1,1 AS A$
30 LSET A$="A"
40 FOR I=1 TO 10:PUT 1:NEXT
50 PRINT LOF(1):CLOSE:STOP
RUN
 2560
Break in 50
Ok
```

dan blijkt dat de lengte van het bestand reeds 5120 bytes is terwijl er effektief pas 10 bytes in zijn geschreven.

De oplossing: om de schijfruimte toch zo efficiënt mogelijk te gebruiken, dienen records met grote recordlengten te worden gebruikt. De slimme programmeur wijst één groot record toe met FIELD en verdeelt deze programma-technisch zelf in kleinere parten.

4.5 Enkele duistere kommando's

De MSX-disk versie bevat enkele kommando's die:

- of nog 'in de steiger' staan en pas in latere versies een definitieve betekenis gaan krijgen

– of in vorige, pre-MSX versies een betekenis hebben gehad en ten onrechte nog niet geheel zijn verwijderd uit het MSX-basic

De volgende kommando's behoren tot die zogenaamde duistere kommando's. Zij zijn in MSX-verband nergens beschreven, maar kunnen in dokumentatie van pre-MSX-computers vaak wel worden gevonden:

SLEUTELWOORD # ATTR$

moeilijkheidsgraad .... **normaal**
soort .... **SYSTEEMVARIABELE**
afkomst .... **ATTR$ is afkorting van attribute string – bijzonderheden (string)**

schrijfwijze

ATTR$(<?>)

Deze systeemvariabele is waarschijnlijk bedoeld om wat algemene gegevens per bestand te weten te komen. Elk gebruik leidt helaas tot een foutmelding.

SLEUTELWOORD # DSKI$

moeilijkheidsgraad . . **zeer moeilijk, kennis van de indeling van de betreffende schijfeenheid is een noodzaak**
soort . . . . A-FUNKTIE
afkomst . . . . **DSKI$ is afkorting van disk input string – invoer string van schijf**

schrijfwijze

```
DSKI$(<SCHIJFNUMMER>,<SECTORNUMMER>)
<SCHIJFNMMER>::=<N>
<SECTORNUMMER>::=<N>
<N>::=<ZIE ALGEMENE SPECIFICATIES>
```

Deze funktie is waarschijnlijk bedoeld om direkt sectoren van schijf te kunnen lezen. Een schijfnummer en een sectornummer dienen dan te worden opgegeven. In werkelijkheid leest de computer de sector wel in maar laat deze in een buffer staan; het kommando

```
LET A$=DSKI$(1,1)
```

heeft tot effect dat de computer wel van schijf leest, maar niets in de variabele A$ zet.

SLEUTELWOORD # DSKO$

moeilijkheidsgraad . . **zeer moeilijk, kennis van de indeling van de betreffende schijfeenheid is een noodzaak**
soort . . . . **KOMMANDO**
afkomst . . . . **DSKO$ is afkorting van disk output string – uitvoer van string naar schijf**

schrijfwijze

DSKO$(<?>)

Dit kommando is waarschijnlijk bedoeld om direkt sectoren op schijf te kunnen zetten. Het kommando bestaat, maar elk gebruik ervan resulteert helaas in foutmeldingen.

SLEUTELWOORD **FPOS**

moeilijkheidsgraad . . zeer moeilijk, kennis van de indeling van de betreffende schijfeenheid is een noodzaak
soort . . . . N-FUNKTIE
afkomst . . . . FPOS is afkorting van file position – positie (van lees/schrijfkop) in bestand

schrijfwijze

FPOS(<?>)

Deze funktie is waarschijnlijk bedoeld om de positie van de lees/schrijf-kop te kunnen achterhalen. Uitvoering leidt altijd tot een foutmelding, meestal een Internal error.

SLEUTELWOORD SET

moeilijkheidsgraad . . . . **normaal**
soort . . . . **KOMMANDO**
afkomst . . . . **SET is klaarzetten**

schrijfwijze

```
SET<?>,<?>
```

Dit kommando is waarschijnlijk bedoeld om bestanden een bepaalde status of bescherming te kunnen geven. Gebruik hiervan leidt echter onherroepelijk tot een foutmelding.

5 FOUTMELDINGEN OP VOLGORDE VAN NUMMER

Het MSX-disk basic kent de volgende foutmeldingen, hier op nummer opgenomen:

01 NEXT without FOR	gepoogd werd om een NEXT uit te voeren zonder dat een bijbehorende FOR werd uitgevoerd.
02 Syntax error	er werd een fout in de schrijfwijze ontdekt.
03 RETURN without GOSUB	gepoogd werd om een RETURN uit te voeren terwijl er niet eerder een GOSUB werd uitgevoerd.
04 Out of DATA	gepoogd werd om een READ uit te voeren terwijl er geen DATA meer kon worden gevonden.
05 Illegal function call	er werd gepoogd om een kommando of funktie uit te voeren met niet toegestane waarden.
06 Overflow	het resultaat van een berekening ligt boven het toegestane maximum of onder het toegestane minimum.
07 Out of memory	er werd gepoogd om een kommando uit te voeren dat meer geheugen nodig heeft dan er beschikbaar is.
08 Undefined line number	een niet bestaand programmaregelnummer werd benaderd.
09 Subscript out of range	een array-variabele werd buiten zijn dimensies aangesproken of er werd een verkeerd aantal dimensies genoemd.
10 Redimensioned array	er werd gepoogd om een array-variabele voor een tweede keer te DIMensioneren.
11 Division by zero	gepoogd werd om een deling door nul te doen of een negatieve macht van nul te bepalen.
12 Illegal direct	er werd gepoogd om een kommando direkt in te tikken terwijl dit kommando slechts in een programmaregel mag voorkomen.

13 Type mismatch	een numerieke en een alfanumerieke uitdrukking werden met elkaar verwisseld.
14 Out of string space	gepoogd werd om een kommando uit te voeren waardoor het stringgeheugen te klein werd. Vergroot het string-geheugen met CLEAR.
15 String too long	er werd gepoogd om een string samen te stellen die langer werd dan 255 posities.
16 String formula too complex	de alfanumerieke uitdrukking die werd uitgewerkt is te ingewikkeld voor MSX-basic. Splits deze uitdrukking in enkele kleinere.
17 Can't continue	gepoogd werd on een programma met CONT te vervolgen terwijl dat niet (meer) gaat.
18 Undefined user function	gepoogd werd om met FN een niet gedefinieerde funktie aan te roepen.
19 Device I/O error	een fout werd ontdekt tijdens het lezen van/schrijven naar een randapparaat (cassetterecorder of printer e.d.).
20 Verify error	tijdens de controle met CLOAD? werd een verschil ontdekt tussen het programma op cassetteband en het programma in het geheugen.
21 No RESUME	de ERROR-routine werd aangeroepen en een RESUME kon niet worden gevonden.
22 RESUME without error	er werd gepoogd om een RESUME uit te voeren terwijl er geen fout via de ON ERROR GOTO werd gedetecteerd.
24 Missing operand	een noodzakelijke uitdrukking wordt in een kommando niet gevonden.
25 Line buffer overflow	een ingetikte programmaregel is te lang voor MSX-basic (langer dan 255 karakters).
50 FIELD overflow	de lengte van het met FIELD gedefinieerde record is groter dan 256 bytes.
51 Internal error	deze melding wijst op een fout die niet zou mogen kunnen voorkomen. Licht uw computerleverancier of MICROSOFT in.

52 Bad file number	een niet toegestaan kanaalnummer werd gebruikt.
53 File not found	het te benaderen bestand werd niet op de schijf aangetroffen.
54 File already open	een reeds bezet kanaal wordt met OPEN een tweede maal gebruikt of een bestand wordt over twee kanalen geopend.
55 Input past end	er werd gepoogd om nog gegevens uit een bestand te lezen terwijl dit bestand reeds volledig was doorgewerkt.
56 Bad file name	een bestandsnaam werd niet volgens de voorschriften samengesteld.
57 Direct statement in file	tijdens het LOADen van een programma werd een direkt kommando (zonder regelnummer) ontdekt; LOAD werd gestopt.
58 Sequential I/O only	een randapparaat dat alleen sequentiele benadering toelaat werd met GET/PUT benaderd.
59 File not open	een kanaal werd aangesproken zonder dat hierop een bestand geOPENd is.
60 Bad FAT	deze foutmelding is wel aanwezig maar kan in het MSX-disk-basic niet voorkomen tenzij met ERROR 60 gegenereerd.
61 Bad file mode	het benaderde bestand is van een verkeerde soort (sequentieel terwijl random werd verwacht of andersom).
62 Bad drive name	er werd een niet bestaande schijveneenheid aangeroepen.
63 Bad sector number	deze foutmelding is wel aanwezig maar kan in het MSX-disk-basic niet voorkomen tenzij met ERROR 63 gegenereerd.
64 File still open	met NAME/COPY/KILL werd een bestand benaderd dat nog is geopend (zie OPEN).
65 File already exists	met NAME werd geprobeerd om de naam van een reeds op de schijf aanwezig bestand aan een ander bestand toe te kennen.
66 Disk full	er is niet genoeg ruimte op de schijf over.

67 Too many files	er werd geprobeerd om een honderddertiende file op een schijf toe te wijzen.
68 Disk write protected	er werd geprobeerd om te schrijven naar een schijf die WRITE PROTECTED is (zie H.3).
69 Disk I/O error	er werd een lees/schrijffout op schijf ontdekt. de schijveneenheid kan iets mankeren of de schijf is niet geinitialiseerd.
70 Disk offline	de schijveneenheid staat niet aangeschakeld of er is geen schijf geplaatst.
71 Rename across disk	er wordt geprobeerd om een bestand met NAME een andere schijveneenheid toe te kennen.

6 FOUTMELDINGEN OP ALFABETISCHE VOLGORDE

Het MSX-disk basic kent de volgende foutmeldingen, hier op alfabetische volgorde opgenomen:

60 Bad FAT	deze foutmelding is wel aanwezig maar kan in het MSX-disk-basic niet voorkomen tenzij met ERROR 60 gegenereerd.
62 Bad drive name	er werd een niet bestaande schijveneenheid aangeroepen.
61 Bad file mode	het benaderde bestand is van een verkeerde soort (sequentieel terwijl random werd verwacht of andersom).
56 Bad file name	een bestandsnaam werd niet volgens de voorschriften samengesteld.
52 Bad file number	een niet toegestaan kanaalnummer werd gebruikt.
63 Bad sector number	deze foutmelding is wel aanwezig maar kan in het MSX-disk-basic niet voorkomen tenzij met ERROR 63 gegenereerd.
17 Can't continue	gepoogd werd om een programma met CONT te vervolgen terwijl dat niet (meer) gaat.
19 Device I/O error	een fout werd ontdekt tijdens het lezen van/schrijven naar een randapparaat (cassetterecorder of printer e.d.).
57 Direct statement in file	tijdens het LOADen van een programma werd een direkt kommando (zonder regelnummer) ontdekt; LOAD werd gestopt.
69 Disk I/O error	er werd een lees/schrijffout op schijf ontdekt. de schijveneenheid kan iets mankeren of de schijf is niet geinitialiseerd.
66 Disk full	er is niet genoeg ruimte op de schijf over.
70 Disk offline	de schijveneenheid staat niet aangeschakeld of er is geen schijf geplaatst.

68 Disk write protected	er werd geprobeerd om te schrijven naar een schijf die WRITE PROTECTED is (zie H.3).
11 Division by zero	gepoogd werd om een deling door nul te doen of een negatieve macht van nul te bepalen.
50 FIELD overflow	de lengte van het met FIELD gedefinieerde record is groter dan 256 bytes.
65 File already exists	met NAME werd geprobeerd om de naam van een reeds op de schijf aanwezig bestand aan een ander bestand toe te kennen.
54 File already open	een reeds bezet kanaal wordt met OPEN een tweede maal gebruikt of een bestand wordt over twee kanalen geopend.
53 File not found	het te benaderen bestand werd niet op de schijf aangetroffen.
59 File not open	een kanaal werd aangesproken zonder dat hierop een bestand geOPENd is.
64 File still open	met NAME/COPY/KILL werd een bestand benaderd dat nog is geopend (zie OPEN).
12 Illegal direct	er werd gepoogd om een kommando direkt in te tikken terwijl dit kommando slechts in een programmaregel mag voorkomen.
05 Illegal function call	er werd gepoogd om een kommando of funktie uit te voeren met niet toegestane waarden.
55 Input past end	er werd gepoogd om nog gegevens uit een bestand te lezen terwijl dit bestand reeds volledig was doorgewerkt.
51 Internal error	deze melding wijst op een fout die niet zou mogen kunnen voorkomen. Licht uw computerleverancier of MICROSOFT in.
25 Line buffer overflow	een ingetikte programmaregel is te lang voor MSX-basic (langer dan 255 karakters).
24 Missing operand	een noodzakelijke uitdrukking wordt in een kommando niet gevonden.

01 NEXT without FOR	gepoogd werd om een NEXT uit te voeren zonder dat een bijbehorende FOR werd uitgevoerd.
21 No RESUME	de ERROR-routine werd aangeroepen en een RESUME kon niet worden gevonden.
04 Out of DATA	gepoogd werd om een READ uit te voeren terwijl er geen DATA meer kon worden gevonden.
07 Out of memory	er werd gepoogd om een kommando uit te voeren dat meer geheugen nodig heeft dan er beschikbaar is.
14 Out of string space	gepoogd werd om een kommando uit te voeren waardoor het stringgeheugen te klein werd. Vergroot het string-geheugen met CLEAR.
06 Overflow	het resultaat van een berekening ligt boven het toegestane maximum of onder het toegestane minimum.
22 RESUME without error	er werd gepoogd om een RESUME uit te voeren terwijl er geen fout via de ON ERROR GOTO werd gedetecteerd.
03 RETURN without GOSUB	gepoogd werd om een RETURN uit te voeren terwijl er niet eerder een GOSUB werd uitgevoerd.
10 Redimensioned array	er werd gepoogd om een arrayvariabele voor een tweede keer te DIMensioneren.
71 Rename across disk	er wordt geprobeerd om een bestand met NAME een andere schijveneenheid toe te kennen.
58 Sequential I/O only	een randapparaat dat alleen sequentiele benadering toelaat werd met GET/PUT benaderd.
16 String formula too complex	de alfanumerieke uitdrukking die werd uitgewerkt is te ingewikkeld voor MSX-basic. Splits deze uitdrukking in enkele kleinere.
15 String too long	er werd gepoogd om een string samen te stellen die langer werd dan 255 posities.

09 Subscript out of range	een array-variabele werd buiten zijn dimensies aangesproken of er werd een verkeerd aantal dimensies genoemd.
02 Syntax error	er werd een fout in de schrijfwijze ontdekt.
67 Too many files	er werd geprobeerd om een honderddertiende file op een schijf toe te wijzen.
13 Type mismatch	een numerieke en een alfanumerieke uitdrukking werden met elkaar verwisseld.
08 Undefined line number	een niet bestaand programmaregelnummer werd benaderd.
18 Undefined user function	gepoogd werd om met FN een niet gedefinieerde funktie aan te roepen.
20 Verify error	tijdens de controle met CLOAD? werd een verschil ontdekt tussen het programma op cassetteband en het programma in het geheugen.

7 FRAME, EEN BASISPROGRAMMA VOOR BESTANDSONDERHOUD

In dit hoofdstuk is de lijst opgenomen van het programma FRAME.

Het programma FRAME is een basisprogramma waarvan iedereen binnen enkele minuten zijn eigen onderhoudsprogramma's kan maken.

Om een eigen bestandsonderhoudprogramma samen te stellen, behoeft men slechts de volgende handelingen te verrichten:

1. Laad het programma FRAME (of tik het de eerste keer in).
2. Breng op regel 40 in een DATA-kommando de naam van het te onderhouden bestand in.
3. Neem op regel 1000 in een DATA-kommando het aantal records in dat het bestand dient te bevatten.
4. Breng op regel 250-490 de gegevens per veld in. Per veld steeds de veldnaam in een DATA-kommando inbrengen, gevolgd door de maximale lengte van het veld.
5. Test het programma uit. Het kan zijn dat de recordlengte groter is dan 256 bytes of dat een veldomschrijving te lang is; kortom, er kunnen kleine schoonheidsfoutjes optreden.
6. Leg het programma tenslotte op schijf vast.

In de lijst zijn wat voorbeeldgegevens op de in te vullen regelnummers opgenomen; deze mogen natuurlijk worden vervangen of verwijderd.

Vanaf regel 60000 is een ingaveroutine opgenomen die fool-proof is. Ook in vele andere programma's kan deze routine zijn nut bewijzen.

De handige programmeur kan het programma misschien later aanpassen, zodat er ook een sorteer- en printmogelijkheid is. Terwille van de eenvoud zijn deze mogelijkheden achterwege gelaten.

Het programma FRAME is in een TOP-DOWN struktuur opgebouwd en doorspekt met dokumentatie. De iets meer geoefende amateur

moet de werking van dit programma toch wel grotendeels kunnen doorgronden.

De werking van FRAME is vrij vanzelfsprekend. Steeds dient éérst een recordnummer te worden ingegeven, gevolgd door de betreffende gegevens.
Onthoud echter het volgende:

Ingave van	resultaat/funktie
RETURN	de inhoud van een veld blijft onveranderd
– (min-toets gevolgd door return)	ga terug naar de vorige ingave
– – (twee maal min)	maak het veld schoon
– – – (drie maal min)	verwijder het record

Het programma FRAME mag op non-commerciële basis worden gekopiëerd ten behoeven van vrienden, kennissen enzovoorts. Het is echter verboden om dit programma op commerciële basis te verhandelen of zonder uitdrukkelijke en schriftelijke toestemming van de uitgever te publiceren.

```
10 REM *************************
20 REM * ALGEMEEN BESTANDSVER- *
30 REM * ZORGINGSPROGRAMMA     *
40 REM * VOOR EEN MSX-COMPUTER *
50 REM * MET FLOPPY DISK.      *
60 REM * (C)1985 STARK TEXEL   *
70 REM *************************
80 REM
90 CLEAR 4096
100 REM ------------------------
110 REM NEEM OP REGEL 140 OP:
120 REM 140 DATA "<file>"
130 REM ------------------------
```

```
140 DATA "KENNIS"
150 REM ------------------------
160 REM NEEM OP REGEL 190 OP:
170 REM 190 DATA <aantal records>
180 REM ------------------------
190 DATA 1000
200 REM ------------------------
210 REM NEEM OP REGEL 250 EN VER-
220 REM DER OP (MAX. 20 KEER):
230 REM XXX DATA "<veld>",<lang>
240 REM ------------------------
250 DATA "NAAM",24
260 DATA "STRAAT",24
270 DATA "POSTKODE",6
280 DATA "WOONPLAATS",24
290 DATA "TELEFOON",15
300 DATA "GEBOORTEDATUM",8
310 DATA "HOBBIES",24
320 DATA "LEEFTIJD",2
330 DATA "LIEVELINGSKLEUR",8
340 DATA "LIEVELINGSDRANK",12
350 DATA "LAATSTE VISITE",8
360 DATA "OPMERKINGEN",24
500 REM ------------------------
510 DATA ""
520 REM *************************
530 REM * HIER BEGINT HET FEI-  *
540 REM * TELIJKE PROGRAMMA PAS *
550 REM *************************
560 REM
570 REM *************************
580 REM *     HOOFDPROGRAMA     *
590 REM *     ------------      *
600 REM * VANUIT DIT HOOFDPRO-  *
610 REM * GRAMMA WORDEN ALLE    *
620 REM * FUNKTIES AANGEROEPEN  *
```

```
625 REM ***************************
630 REM
640 GOSUB 1000'INITIALISATIE
650 GOSUB 1500'PROGRAMMAKEUZE
660 REM NAAR DE DRIE PROGRAMMA'S
670 ON K GOSUB 2000,3000,4000
680 GOTO 650' EN WEER PROGRAMMAKEUZE
1000 REM ***************************
1010 REM *     INITIALISATIE       *
1020 REM *     -------------       *
1030 REM * HIER WORDEN DE VASTE    *
1040 REM * WAARDEN BEPAALD EN      *
1050 REM * HET BESTAND TOEGEWE-    *
1060 REM * ZEN OF GEOPEND.         *
1070 REM ***************************
1080 REM
1090 REM BESTAND TOEWIJZEN
1100 RESTORE:READ F$,RM
1110 OPEN F$ AS #1
1120 REM VELDOMSCHRIJVINGEN BEPALEN
1130 DIM V$(21),L(20),VI$(20)
1140 RL=0:FOR I=1 TO 20:READ V$(I)
1150 IF V$(I)<>"" THEN READ L(I):RL=RL+L
(I):NEXT I
1160 REM RECORD TOEWIJZEN
1170 FIELD #1,RL+1 AS R$
1180 REM INPUT PAST END ERROR TE
1190 REM SLIM AF ZIJN
1200 PUT #1,RM+1
1210 RETURN
1500 REM ***************************
1510 REM *    PROGRAMMAKEUZE       *
1520 REM *    --------------       *
1530 REM * HIER WORDT DE PRO-      *
1540 REM * RAMMAKEUZE GEREGELD     *
1550 REM ***************************
```

```
1560 REM
1570 K$="PROGRAMMAKEUZE"
1580 GOSUB 55000'CLS/KOP
1590 LOCATE 0,3
1610 PRINT "1=AANMAKEN/WIJZIGEN RECORDS"
1620 PRINT "2=OPVRAGEN RECORDS"
1630 PRINT "3=EINDE PROGRAMMA"
1640 I$="000901UW KEUZE"
1650 GOSUB 60000'INPUT
1660 IF II$<>"1" AND II$<>"2" AND II$<>"
3" THEN 1650'FOUT
1670 K=VAL(II$):RETURN
2000 REM ***************************
2010 REM *   AANMAKEN/WIJZIGEN     *
2020 REM *   -----------------     *
2030 REM * HIER WORDT DE AANMAAK *
2040 REM * EN WIJZIGING OF VER-  *
2050 REM * WIJDERING VAN DE RE-  *
2060 REM * CORDS GEREGELD.         *
2070 REM ***************************
2080 REM
2090 K$="AANMAKEN/WIJZIGEN RECORDS"
2100 GOSUB 55000'CLS/KOP
2110 GOSUB 50000'PRINT VELDEN
2120 GOSUB 45000'INGAVE/LEES RECORD
2125 IF RN=0 THEN RETURN'0=EINDE
2130 REM NU INGAVE VELDEN
2140 FOR I=1 TO 20:IF V$(I)<>"" THEN I$=
RIGHT$(STR$(MV+101),2)+RIGHT$(STR$(I+103
),2)+RIGHT$(STR$(L(I)+100),2):GOSUB 6000
0 ELSE 2170'INPUT
2150 IF II$=STRING$(L(I)," ") THEN II$=V
I$(I)
2155 IF LEFT$(II$,3)="---" THEN LSET R$=
"":PUT #1,RN:GOTO 2100
2156 IF LEFT$(II$,2)="--" THEN II$=SPACE
```

```
$(L(I)):GOTO 2160
2157 IF LEFT$(II$,1)="-" THEN LOCATE MV+
1,I+3:PRINT VI$(I);:I=I-2:GOTO 2165
2160 VI$(I)=II$:LOCATE MV+1,I+3:PRINT II
$;
2165 IFI=-1 THEN I=0
2166 NEXT I
2170 GOSUB 40000'SCHRIJF RECORD
2180 GOTO2120' VOLGENDE RECORD
3000 REM ***************************
3010 REM *    OPVRAGEN RECORDS     *
3020 REM *    ----------------     *
3030 REM * HIER WORDEN DE RE-      *
3040 REM * CORDS OPGEVRAAGD.       *
3050 REM ***************************
3060 REM
3070 K$="OPVRAGEN RECORDS"
3080 GOSUB 55000'CLS/KOP
3090 GOSUB 50000'PRINT VELDEN
3100 GOSUB 45000'INGAVE/LEES RECORD
3110 IF RN=0 THEN RETURN ELSE 3100
4000 REM ***************************
4010 REM *    EINDE PROGRAMMA      *
4020 REM ***************************
4030 REM
4040 CLOSE:CLS:STOP
40000 REM ***************************
40010 REM *      SCHRIJF RECORD     *
40020 REM *      --------------     *
40030 REM * HIER WORDT HET EERDER *
40040 REM * INGELEZEN RECORD WEER *
40050 REM * GESCHREVEN OP SCHIJF. *
40060 REM ***************************
40070 REM
40080 RR$="*":FOR I=1 TO 20:IF V$(I)<>""
 THEN RR$=RR$+VI$(I):NEXT I
```

```
40090 LSET R$=RR$:PUT #1,RN:RETURN
45000 REM *************************
45010 REM *  INGAVE RECORDNUMMER  *
45020 REM *  -----------------    *
45030 REM * HIER WORDT HET RE-    *
45040 REM * CORDNUMMER INGEGEVEN  *
45050 REM * EN HET RECORD INGELE- *
45060 REM * ZEN EN AFGEDRUKT.     *
45070 REM *************************
45080 REM
45090 I$="000304RECORD"+STRING$(MV-6," "
)
45100 GOSUB 60000'INPUT
45110 FOR I=1 TO 4:IF (MID$(II$,I,1)<"0"
 OR MID$(II$,I,1)>"9") AND MID$(II$,I,1)
<>" " THEN 45100 ELSE NEXT I
45120 RN=VAL(II$):IF RN=0 THEN 45190
45125 IF RN>RM THEN 45090
45126 LOCATE MV+1,3:PRINT USING"      ###
#";RN
45130 GET #1,RN:IF LEFT$(R$,1)<>"*" THEN
 LSET R$="*"
45140 RP=2:FOR I=1 TO 20:IF V$(I)="" THE
N 45160
45150 VI$(I)=MID$(R$,RP,L(I)):RP=RP+L(I)
:NEXT I
45160 FOR I=1 TO 20:IF V$(I)="" THEN 451
90
45170 LOCATE MV+1,I+3:PRINT VI$(I);:NEXT
 I
45190 RETURN
50000 REM *************************
50010 REM *      PRINT VELDEN     *
50020 REM *      ----------       *
50030 REM * HIER WORDEN DE VELD-  *
50040 REM * NAMEN AFGEDRUKT.      *
```

```
50050 REM *************************
50060 REM
50070 LOCATE 0,3:MV=6:PRINT "RECORD"
50080 FOR I=1 TO 20
50090 IF V$(I)="" THEN 50120
50100 LOCATE 0,I+3:PRINT V$(I);:IF MV<LE
N(V$(I)) THEN MV=LEN(V$(I))
50110 NEXT I
50120 FOR I=1 TO 21:LOCATE MV,I+2:PRINT"
:";:IF V$(I)<>"" THEN NEXT I
50130 RETURN
55000 REM *************************
55010 REM *  BEELD SCHOON EN KOP  *
55020 REM *  -----------------    *
55030 REM * HIER WORDT HET BEELD  *
55040 REM * SCHOONGEMAAKT EN DE   *
55050 REM * KOP AFGEDRUKT         *
55060 REM *************************
55070 REM
55080 KEY OFF:COLOR 15,4,4:WIDTH 40:SCRE
EN 0:CLS
55090 PRINT STRING$(40,"█")
55100 LOCATE (40-LEN(K$))/2-1,0:PRINT "
";K$;" "
55110 RETURN
60000 REM *************************
60010 REM *    INGAVEROUTINE      *
60020 REM *    ------------       *
60030 REM * I$="HHVVTTA---A"      *
60040 REM * HH=HORIZONTALE POS    *
60050 REM * VV=VERTIKALE POS.     *
60060 REM * TT=AATAL TEKENS.      *
60070 REM * A---A=TEKST INPUT     *
60080 REM * RESULTAAT, AANGEVULD  *
60090 REM * MET SPATIES IN II$    *
60100 REM *************************
```

```
60110 REM
60120 II$="": XX=VAL(MID$(I$,1,2)):YY=VA
L(MID$(I$,3,2)):TT=VAL(MID$(I$,5,2)):SS$
=MID$(I$,7):IF LEN(SS$) THEN SS$=SS$+":"
60130 LOCATE XX,YY,0:PRINT SS$;STRING$(T
T,"▬");:LOCATE XX+LEN(SS$),YY
60140 KK$=INKEY$:IF KK$="" THEN 60140
60150 IF KK$=CHR$(8) OR ASC(KK$)=127 OR
ASC(KK$)=29 THEN 60210
60160 IF KK$=CHR$(13) THEN 60240
60170 IF ASC(KK$)<32 OR ASC(KK$)>122 THE
N BEEP:GOTO 60140
60180 IF ASC(KK$)>96 AND ASC(KK$)<123 TH
EN LET KK$=CHR$(ASC(KK$)-32)
60190 IF LEN(II$)=TT THEN BEEP:GOTO 6014
0
60200 II$=II$+KK$:PRINT KK$;:GOTO 60140
60210 IF II$="" THEN BEEP:GOTO 60140
60220 II$=LEFT$(II$,LEN(II$)-1):PRINT CH
R$(8);"ò";CHR$(8);:GOTO 60140
60240 II$=II$+STRING$(TT-LEN(II$),32):LO
CATE XX+LEN(SS$),YY:PRINT II$;:RETURN
60250 RETURN
```

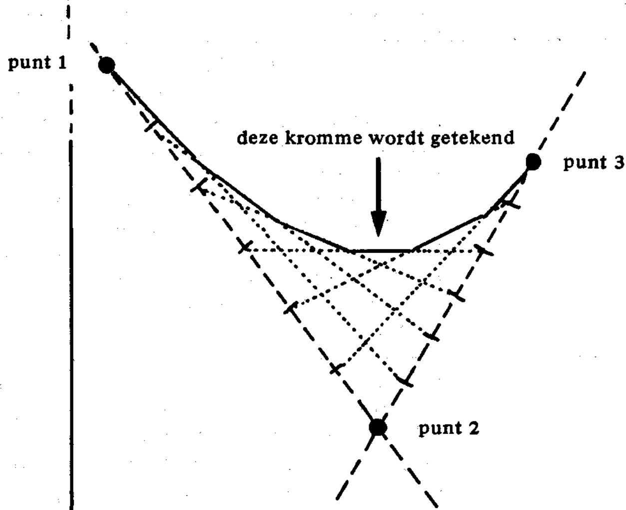
In dit hoofdstuk is de lijst opgenomen van het programma TEKEN.

Het programma TEKEN is een vrij uitgebreid en geavanceerd tekentafelprogramma; technische tekeningen kunnen worden ingebracht, vergroot, verkleind, verschoven en kunnen op schijf worden opgeslagen.

Nadat het programma is opgestart, wordt een leeg beeldscherm gepresenteerd. De volgende kommando's zijn mogelijk;

TOETS	FUNKTIE (zorg dat CAOS LOCK aan staat)
pijl omhoog	verplaatst de tekenpijl naar boven.
pijl naar rechts	verplaatst de tekenpijl naar rechts.
pijl naar beneden	verplaatst de tekenpijl naar beneden.
pijl naar links	verplaatst de tekenpijl naar links.
–	(mintoets) verbetert de laatste aktie.
S	vergroot/verkleint de 'stappen' die de tekenpijl tegelijk over het beeldscherm neemt van 1 naar 5 puntjes of van 5 naar 1 puntje.
R	maakt de tekening opnieuw maar nu met een tekenraster er overheen. Met de R kan dit raster ook weer worden uitgeschakeld.
spatie	met de spatie kan een punt worden geplaatst ter hoogte van de tekenpijl. Deze punt dient om de lokatie voor een latere aktie vast te leggen.
L	trekt een lijn tussen de twee laatste (met spatie) geplaatste punten. Na L blijft alleen de laatste punt aktief; de andere verdwijnen.
C	tekent een cirkel. De laatste twee geplaatste

	punten vormen twee middellijnpunten op de cirkel. Het laatst geplaatste punt blijft aktief, het andere andere wordt verwijderd.
B	tekent een rechthoek. Het eerste geplaatste puntje en het tweede geplaatste puntje vormen de hoeken van een diagonaal van de rechthoek. Alleen het laatste puntje blijft aktief.
P	start veelhoek. Nadat met P een punt geplaatst is, kunnen diverse verbindingslijnen worden getekend om bijvoorbeeld een veelhoek te tekenen. De figuur kan als laatste aktie dan worden gesloten met de D-funktie.
D	maak veelhoek dicht. Het laatste aktieve punt wordt met het eerste via P geplaatste punt verbonden.
Q	teken kromme. Voor deze funktie dienen *drie* punten met spatie op het beeldscherm te zijn geplaatst. De wijze waarop de kromme wordt getekend, laat zich het best in een tekening verduidelijken.



	Op deze wijze kunnen allerlei krommen, ook ellipsen, cirkeldelen etcetera, worden geconstrueerd.
W...	na de letter W dient een cijfer (0 ... 9) of een letter (A ... Z) te worden ingetoetst. Indien deze intoetsing langer dan een seconde op zich laat wachten, wordt deze funktie niet uitgevoerd. De tekening wordt in bestand "TEK....DAT" (op ... staat het ingetoetste teken) opgeslagen.
O...	na de letter O dient net zoals bij het kommando -W- een letter of cijfer te worden ingetoetst. De eerder in "TEK....DAT" opgeslagen tekening wordt weer opgehaald.
A...	na de letter A dient net zoals bij het kommando -W- een letter of cijfer te worden ingetoetst. De eerder in "TEK....DAT" opgeslagen tekening wordt vanaf het laatst geplaatste punt opgehaald en *bij* de reeds aanwezige tekening gevoegd. Zo kunt u een éénmaal getekend detail steeds weer opnieuw, desnoods meerdere malen, automatisch in een tekening bijplaatsen.
X	maakt de gehele aktieve tekening schoon.
G	vergroot de tekening met een faktor twee. Deze funktie kan maximaal drie maal achter elkaar worden gebruikt (8 maal vergroten). Het laatst geplaatste punt blijft op zijn plaats, de rest van de tekening wordt hier omheen uitvergroot. In de vergrote toestand kunnen veranderingen in de tekening worden aangebracht die later bij het verkleinen ook weer worden meeverkleind. Zo kunnen detailtekeningen eerst groot worden ingevoerd en pas later naar de werkelijke grootte worden verkleind.
K	verkleint de tekening met een faktor twee. Deze funktie kan alleen worden gebruikt na een vergroting en brengt de tekening weer naar een eerder, kleiner formaat. Het laatst geplaatste punt blijft op zijn plaats; de rest van de tekening wordt hier om-

	heen verkleind.
V	verschuift de tekening. De gehele tekening wordt verschoven zover als het eerst en het tweede puntje uit elkaar liggen. Zo kan een afgewerkt deel van de tekening (ook in vergrote toestand) uit beeld worden geschoven en kan daarnaast worden verder getekend; het beeldscherm maakt steeds slechts een gedeelte zichtbaar. De V-funktie maakt het mogelijk om tekeningen te maken die véél groter dan het beeldscherm zijn. Het beeldscherm vormt slechts een soort 'venster' waardoor een gedeelte van een tekening zichtbaar is. Zo kan een tekening met een bereik van meer dan 30 000 x 30 000 beeldschermpuntjes worden gemaakt.
F	kleurt een gedeelte wit in. Vanuit het laatst getekende puntje wordt een met lijnen omsloten oppervlak wit ingekleurd. De maximale ingewikkeldheid van een tekening is beperkt tot 500 akties (lijnen, krommen, cirkels, inkleuringen of blokken). Binnen dit aantal akties kunnen over het algemeen vrij gedetailleerde tekeningen ruim worden gekodeerd. *Let op: SHIFT LOCK moet bij dit programma altijd AAN staan.*

Het programma TEKEN is zo ontworpen dat de handige programmeur zonder veel moeite een aanpassing kan maken voor het aansluiten van een plotter.

Het programma TEKEN is in een TOP-DOWN struktuur geschreven. De berekeningen en gedachtengangen die achter het programma zitten, maken het ondanks alle kommentaar en de duidelijke opbouw toch een moeilijk geheel. Een hele slimme amateur doorgrondt het programma op den duur misschien helemaal.

Het werken met dit programma is een erg leuke ervaring. Al snel bent u in staat om mooie tekeningen te fabriceren.

Het programma TEKEN mag op non-commerciële basis worden geko-

piëerd ten behoeve van vrienden, kennissen enzovoorts. Het is echter verboden om dit programma op commerciële basis te verhandelen of zonder uitdrukkelijke en schriftelijke toestemming van de uitgever te publiceren.

```
10 REM **************************
20 REM *    TEKENPROGRAMMA      *
30 REM *    --------------      *
40 REM *  (C)1985 STARK TEXEL   *
50 REM **************************
60 REM
70 REM **************************
80 REM *    HOOFDPROGRAMMA      *
90 REM *    --------------      *
100 REM**************************
110 REM
120 GOSUB 150'INITIALISATIE
130 GOSUB 250'KOMMANDO INTEPRETATOR
140 STOP
150 REM **************************
160 REM *     INITIALISATIE      *
170 REM *     -------------      *
180 REM **************************
190 REM
200 COLOR 15,4,4:SCREEN 2,0
210 DIM P%(500,6),X%(2),Y%(2)
220 SPRITE$(1)=CHR$(128):SPRITE$(2)=CH
(128):SPRITE$(3)=CHR$(128):SPRITE$(0)=
R$(240)+CHR$(192)+CHR$(160)+CHR$(144):
SUB 3070'PIJL
230 GOSUB 1620'SCHOONMAKEN TEKENING
240 RETURN
250 REM **************************
260 REM * KOMMANDO INTEPRETATOR *
270 REM * --------------------- *
280 REM * HIER WORDEN DE TOETS- *
```

```
290 REM * INSLAGEN GECONTRO-     *
300 REM * LEERD EN DE BIJBEHO-   *
310 REM * RENDE ACTIES ONDERNO-  *
320 REM * MEN.                   *
330 REM *************************
340 REM
350 C$=CHR$(30)+CHR$(28)+CHR$(31)+CHR$
9)+"-SR LCBOWXGKVFDPAQ"
360 K$=INKEY$:IF K$="" THEN 360
370 FOR I%=1 TO LEN(C$):IF MID$(C$,I%,
=K$ THEN 380 ELSE NEXT I%:GOTO 360
380 ON I% GOSUB 590,660,730,800,870,95
1010,1080,1170,1240,1310,1380,1510,162
1690,1750,1810,1890,1960,2030,2090,223
390 GOSUB 3070'PIJL
400 GOTO 360
410 REM *************************
420 REM *      CLS/KOP/GRID      *
430 REM *      ------------      *
440 REM * HIER WORDT HET BEELD-  *
450 REM * SCHERM SCHOONGEMAAKT,  *
460 REM * KOPREGEL GESCHREVEN    *
470 REM * EN EVENTUEEL EEN RAS-  *
480 REM * TER GETEKEND BIJ GR=1  *
490 REM *************************
500 REM
510 CLS
520 LINE (10,10)-(240,10):LINE -(240,180
):LINE -(10,180):LINE -(10,10)
530 PAINT(0,0):GOSUB 3000'KOPTEKST
540 IF GR%=0 THEN 580
550 FOR I%=20 TO 240 STEP 10*VF%
560 LINE (I%,10)-(I%,180),3:IF I%<180 TH
EN LINE (10,I%)-(240,I%),3
570 NEXT I%
580 CLOSE:RETURN
```

```
590 REM *************************
600 REM *      PIJL OMHOOG      *
610 REM *      -----------      *
620 REM *************************
630 REM
640 Y%=Y%-ST%:IF Y%<-VY% THEN Y%=Y%+ST%
650 RETURN
660 REM *************************
670 REM *   PIJL NAAR RECHTS    *
680 REM *   ----------------    *
690 REM *************************
700 REM
710 X%=X%+ST%:IF X%>230/VF%-VX% THEN X%=
X%-ST%
720 RETURN
730 REM *************************
740 REM *   PIJL NAAR BENEDEN   *
750 REM *   -----------------   *
760 REM *************************
770 REM
780 Y%=Y%+ST%:IF Y%>170/VF%-VY% THEN Y%=
Y%-ST%
790 RETURN
800 REM *************************
810 REM *   PIJL NAAR LINKS     *
820 REM *   ---------------     *
830 REM *************************
840 REM
850 X%=X%-ST%:IF X%<-VX% THEN X%=X%+ST%
860 RETURN
870 REM *************************
880 REM *  MINTOETS=CORRECTIE   *
890 REM *  ------------------   *
900 REM *************************
910 REM
920 IF PP%=0 THEN RETURN ELSE PP%=PP%-1
```

```
930 GOSUB 2300'TEKENING
940 RETURN
950 REM *************************
960 REM * S-TOETS SNEL/LANGZAAM *
970 REM * --------------------- *
980 REM *************************
990 REM
1000 ST%=1-4*(ST%=1):RETURN
1010 REM *************************
1020 REM * R-TOETS RASTER AAN/UIT *
1030 REM * ---------------------- *
1040 REM *************************
1050 REM
1060 GR%=NOT(GR%):GOSUB 2300'HERTEKENEN
1070 RETURN
1080 REM *************************
1090 REM * SPATIE=PUNT PLAATSEN  *
1100 REM * --------------------  *
1110 REM *************************
1120 REM
1130 X%(0)=X%(1):Y%(0)=Y%(1):X%(1)=X%(2)
:Y%(1)=Y%(2):X%(2)=X%:Y%(2)=Y%
1140 PN%=PN%+1+3*(PN%=3):PUTSPRITE PN%,(
(X%+VX%)*VF%+10,(Y%+VY%)*VF%+9),1
1150 IF SP%=0 THEN SP%=1:SX%=X%:SY%=Y%
1160 P%=P%-(P%<3):RETURN
1170 REM *************************
1180 REM *    L=LIJN TREKKEN     *
1190 REM *    --------------     *
1200 REM *************************
1210 REM
1220 E%=2:I%=0:GOSUB 3130'PLAATS IN P%
1230 RETURN
1240 REM *************************
1250 REM *   C=CIRCEL TEKENEN    *
1260 REM *   ----------------    *
```

```
1270 REM ************************
1280 REM
1290 E%=2:I%=1:GOSUB 3130'PLAATS IN P%
1300 RETURN
1310 REM ************************
1320 REM *     B=BLOK TEKENEN      *
1330 REM *     --------------      *
1340 REM ************************
1350 REM
1360 E%=2:I%=2:GOSUB 3130'PLAATS IN P%
1370 RETURN
1380 REM ************************
1390 REM *  O=OPHALEN TEKENING    *
1400 REM *  ------------------    *
1410 REM ************************
1420 REM
1430 GOSUB 3220'BESTANDSNAAM
1440 IF F$="" THEN RETURN
1450 GOSUB 1620'SCHOONMAKEN
1460 OPEN F$ FOR INPUT AS 1:INPUT #1,R$:
J%=VAL(R$)
1470 FOR PP%=0 TO J%-1:FOR I%=0 TO 6:INP
UT#1,R$:P%(PP%,I%)=VAL(R$):NEXT I%:GOSUB
 2410'TEKEN EEN DETAIL
1480 NEXT PP%:CLOSE
1490 GOSUB 3000'KOPTEKST
1500 RETURN
1510 REM ************************
1520 REM *  W=WEGSCHRIJVEN TEK.   *
1530 REM *  -------------------   *
1540 REM ************************
1550 REM
1560 GOSUB 3220'BESTANDSNAAM
1570 IF F$="" THEN RETURN
1580 OPEN F$ FOR OUTPUT AS 1:R$=STR$(PP%
):PRINT #1,R$
```

```
1590 FOR J%=0 TO PP%-1:FOR I%=0 TO 6
1600 R$=STR$(P%(J%,I%)):PRINT #1,R$
1610 NEXT I%:NEXT J%:CLOSE:RETURN
1620 REM **************************
1630 REM *  X=SCHOONMAKEN TEK.     *
1640 REM *  ------------------     *
1650 REM **************************
1660 REM
1670 SP%=0:PP%=0:VX%=0:VY%=0:VF%=1:X%=0:
Y%=0:P%=0:PN%=0:ST%=1:PUTSPRITE 1,(-1,-1
):PUTSPRITE 2,(-1,-1):PUTSPRITE 3,(-1,-1
):GOSUB 2300:RETURN
1680 PUTSPRITE 0,(10,10),12:RETURN
1690 REM **************************
1700 REM *  G=VERGROOT MAAL TWEE *
1710 REM *  -------------------- *
1720 REM **************************
1730 REM
1740 IF VF%=8 OR P%=0 THEN RETURN ELSE V
F%=VF%*2:VX%=(VX%-X%(2))/2:VY%=(VY%-Y%(2
))/2:P%=0:SP%=0:GOSUB 2300:PUTSPRITE 1,(
-1,-1):PUTSPRITE2,(-1,-1):PUTSPRITE 3,(-
1,-1):RETURN
1750 REM **************************
1760 REM * V=VERKLEIN MAAL TWEE   *
1770 REM * --------------------   *
1780 REM **************************
1790 REM
1800 IF VF%=1 OR P%=0 THEN RETURN ELSE V
F%=VF%/2:VX%=X%(2)+2*VX%:VY%=Y%(2)+2*VY%
:SP%=0:P%=0:GOSUB 2300:PUTSPRITE 1,(-1,-
1):PUTSPRITE 2,(-1,-1):PUTSPRITE 3,(-1,-
1):RETURN
1810 REM **************************
1820 REM *      V=VERSCHUIF       *
1830 REM *      -----------       *
```

```
1840 REM **************************
1850 REM
1860 IF P%<2 THEN RETURN ELSE VX%=VX%+X%
(2)-X%(1):VY%=VY%+Y%(2)-Y%(1):GOSUB 2300
:PUTSPRITE 1,(-1,-1):PUTSPRITE 2,(-1,-1)
:PUTSPRITE 3,(-1,-1)
1870 X%=X%-X%(2)+X%(1):Y%=Y%-Y%(2)+Y%(1)
:P%=0
1880 RETURN
1890 REM **************************
1900 REM *      F=INKLEUREN       *
1910 REM *      -----------       *
1920 REM **************************
1930 REM
1940 E%=1:I%=3:GOSUB 3130'PLAATS IN P%
1950 RETURN
1960 REM **************************
1970 REM *    D=SLUIT POLYGOON    *
1980 REM *    ----------------    *
1990 REM **************************
2000 REM
2010 E%=1:I%=4:GOSUB 3130' PLAATS IN P%
2020 RETURN
2030 REM **************************
2040 REM *    P=START POLYGOON    *
2050 REM *    ----------------    *
2060 REM **************************
2070 REM
2080 SP%=0:GOSUB 1080:RETURN
2090 REM **************************
2100 REM * A=BIJVOEGEN TEKENING   *
2110 REM * --------------------   *
2120 REM **************************
2130 REM
2140 IF P%=0 THEN RETURN
2150 GOSUB 3220'BESTANDSNAAM
```

```
2160 IF F$="" THEN RETURN
2170 OPEN F$ FOR INPUT AS 1:INPUT #1,R$:
J%=VAL(R$)
2180 FOR PP%=PP% TO PP%+J%-1:FOR I%=0 TO
 6:INPUT#1,R$:P%(PP%,I%)=VAL(R$):NEXT I%
:P%(PP%,0)=P%(PP%,0)+X%(2):P%(PP%,1)=P%(
PP%,1)+Y%(2):P%(PP%,2)=P%(PP%,2)+X%(2):P
%(PP%,3)=P%(PP%,3)+Y%(2)
2190 P%(PP%,4)=P%(PP%,4)+X%(2):P%(PP%,5)
=P%(PP%,5)+Y%(2):GOSUB 2410'TEKEN EEN DE
TAIL
2200 NEXT PP%:CLOSE
2210 GOSUB 3000'KOPTEKST
2220 RETURN
2230 REM ***********************
2240 REM *  K=TEKEN EEN KROMME   *
2250 REM *  ----------------     *
2260 REM ***********************
2270 REM
2280 E%=3:I%=5:GOSUB 3130'PLAATS IN P%
2290 RETURN
2300 REM ***********************
2310 REM *  TEKEN HELE TEKENING  *
2320 REM *  ------------------   *
2330 REM ***********************
2340 REM
2350 GOSUB 410'KOP EN RASTER
2360 IF PP%=0 THEN RETURN
2370 QQ%=PP%:FOR PP%=0 TO QQ%-1
2380 GOSUB 2410'TEKEN 1 FIGUUR
2390 NEXT PP%:PP%=QQ%:GOSUB 3000'KOPTEKS
T
2400 RETURN
2410 REM ***********************
2420 REM *    TEKEN EEN DETAIL   *
2430 REM *    ---------------    *
```

```
2440 REM *************************
2450 REM
2460 ON P%(PP%,6)+1 GOSUB 2480,2540,2600
,2660,2750,2820
2470 RETURN
2480 REM *************************
2490 REM *     TEKEN EEN LIJN      *
2500 REM *     --------------      *
2510 REM *************************
2520 REM
2530 LINE ((P%(PP%,2)+VX%)*VF%+10,(P%(PP
%,3)+VY%)*VF%+10)-((P%(PP%,4)+VX%)*VF%+1
0,(P%(PP%,5)+VY%)*VF%+10):RETURN
2540 REM *************************
2550 REM *    TEKEN EEN CIRCEL     *
2560 REM *    ----------------     *
2570 REM *************************
2580 REM
2590 CIRCLE (((P%(PP%,4)+P%(PP%,2))/2+VX
%)*VF%+10,((P%(PP%,5)+P%(PP%,3))/2+VY%)*
VF%+10),VF%*SQR((P%(PP%,2)-P%(PP%,4))^2+
(P%(PP%,3)-P%(PP%,5))^2)/2:RETURN
2600 REM *************************
2610 REM *     TEKEN EEN BLOK      *
2620 REM *     --------------      *
2630 REM *************************
2640 REM
2650 LINE ((P%(PP%,2)+VX%)*VF%+10,(P%(PP
%,3)+VY%)*VF%+10)-((P%(PP%,4)+VX%)*VF%+1
0,(P%(PP%,5)+VY%)*VF%+10),,B:RETURN
2660 REM *************************
2670 REM *      VUL EEN VLAK       *
2680 REM *      ------------       *
2690 REM *************************
2700 REM
2710 XF%=(P%(PP%,4)+VX%)*VF%+10:YF%=(P%(
```

```
PP%,5)+VY%)*VF%+10
2720 IF XF%<10 OR XF%>240 OR YF%<10 OR Y
F%>180 THEN 2740
2730 PAINT (XF%,YF%)
2740 RETURN
2750 REM *************************
2760 REM *    SLUIT POLYGOON      *
2770 REM *    --------------      *
2780 REM *************************
2790 REM
2800 SP%=0:P%(PP%,2)=SX%:P%(PP%,3)=SY%:P
%(PP%,6)=0:GOSUB 2410'TEKEN EEN DETAIL
2810 RETURN
2820 REM *************************
2830 REM *    KROMME TEKENEN      *
2840 REM *    --------------      *
2850 REM *************************
2860 REM
2870 AX=P%(PP%,0):AY=P%(PP%,1):BX=P%(PP%
,2):BY=P%(PP%,3):CX=P%(PP%,4):CY=P%(PP%,
5):D=(BX-AX)^2+(BY-AY)^2:D1=(CX-BX)^2+(C
Y-BY)^2:IF D>D1 THEN D=D1
2880 IF D=0 THEN 2990 ELSE D=SQR(D)/2
2890 DX=(BX-AX)/D:DY=(BY-AY)/D:EX=(CX-BX
)/D:EY=(CY-BY)/D:BX=BX+EX:BY=BY+EY:PSET
((AX+VX%)*VF%+10,(AY+VY%)*VF%+10):GOTO 2
970
2900 Q1=BX-AX:IF Q1=0 THEN 2960
2910 Q1=(BY-AY)/Q1:Q2=AY-Q1*AX
2920 Q3=BX+EX-AX-DX:IF Q3=0 THEN 2960
2930 Q3=(BY+EY-AY-DY)/Q3:Q4=AY+DY-Q3*(AX
+DX)
2940 IF Q1-Q3=0 THEN 2960 ELSE Q3=(Q4-Q2
)/(Q1-Q3)
2950 LINE -((Q3+VX%+.5)*VF%+10,(Q1*Q3+Q2
+VY%+.5)*VF%+10)
```

```
2960 BX=BX+EX:BY=BY+EY:AX=AX+DX:AY=AY+DY
2970 IF ABS(BX-CX)<=ABS(EX) AND ABS(BY-C
Y)<=ABS(EY) THEN LINE -((CX+VX%)*VF%+10,
(CY+VY%)*VF%+10):GOTO 2990
2980 GOTO 2900
2990 RETURN
3000 REM ************************
3010 REM *       KOPTEKST        *
3020 REM *       --------        *
3030 REM ************************
3040 REM
3050 OPEN "GRP:" AS 1:PRESET(8,2):COLOR
0:PRINT #1,"MSX-TEKENPROGRAMMA STARK TEX
EL":COLOR 15:CLOSE
3060 RETURN
3070 REM ************************
3080 REM *     TEKEN DE PIJL     *
3090 REM *     -------------     *
3100 REM ************************
3110 PUTSPRITE 0,((X%+VX%)*VF%+10,(Y%+VY
%)*VF%+9),14:RETURN
3120 RETURN
3130 REM ************************
3140 REM * PLAATS FIGUUR IN P%   *
3150 REM * -------------------   *
3160 REM ************************
3170 REM
3180 IF P%<E% THEN 3210
3190 P%=1:P%(PP%,0)=X%(0):P%(PP%,1)=Y%(0
):P%(PP%,2)=X%(1):P%(PP%,3)=Y%(1):P%(PP%
,4)=X%(2):P%(PP%,5)=Y%(2):P%(PP%,6)=I%:G
OSUB 2410'TEKEN 1 FIGUUR
3200 PP%=PP%+1:PUTSPRITE PN%+1+3*(PN%=3)
,(-1,-1):PUTSPRITE PN%+2+3*(PN%>1),(-1,-
1):GOSUB 3000'KOPTEKST
3210 RETURN
```

```
3220 REM *************************
3230 REM * SPECIFICEREN TEK.NAAM *
3240 REM * --------------------- *
3250 REM * HIER KAN BINNEN EEN   *
3260 REM * BEPAALDE TIJD EEN 0/9 *
3270 REM * WORDEN INGEGEVEN VOOR *
3280 REM * HET TEKENBESANDNUMMER *
3290 REM *************************
3300 REM
3310 FOR I=1 TO 100:K$=INKEY$:IF K$="" T
HEN NEXT I:F$="":RETURN
3320 IF (K$<"A" OR K$>"Z") AND (K$<"0" O
R K$>"9") THEN RETURN
3330 F$="TEK"+K$+".DAT":RETURN
```

Het volgende voorbeeld laat zien dat in eèn enkel geval de DSKI$-funktie nuttig kan worden gebruikt. Door de onvolledige invoering door Microsoft van dit sleutelwoord is het gebruik echter vrij moeilijk.

Dit nuttige programmaatje geeft van de laatst gebruikte schijveneenheid een inhoudsopgave en zal op de meeste MSX-computers, soms na een kleine aanpassing, goed lopen.

```
10 REM ************************
20 REM *  UITGEBREIDE "FILES"  *
30 REM ************************
40 REM
50 MAXFILES=1:CLEAR 100:COLOR 15,4,4:SCR
EEN 0:CLS
60  PRINT "OVERZICHT  SCHIJF"
70 PRINT "------------------"
80 PRINT "  BESTAND   KB"
90 PRINT "------------------"
100 CLEAR 1024
110 FOR S=5 TO 11
120 A$=DSKI$(0,S)
130 FOR I=60310!TO 60821! STEP 32:IF PEE
K(I)<32 OR PEEK(I)>127 THEN 160
140 F$="":FOR J=0 TO 11:F$=F$+CHR$(PEEK(
J+I)):NEXT J
150 PRINT F$;:OPEN F$ AS 1:K=INT(LOF(1)/
1024+.9999):PRINT USING "### KB";K:C=C+K
:CLOSE
160 NEXT I
170 NEXT S
180 PRINT TAB(11);"---"
190 PRINT "BEZET      :";:PRINT USING "##
# KB ";C:PRINT "VRIJ       :";:PRINT USIN
```

```
G "### KB":DSKF(0)
200 PRINT TAB(11);"---":PRINT"TOTAAL
:";:PRINT USING "### KB";DSKF(0)+C:PRINT
210 STOP
```

10 EEN SNEL KOPIEERPROGRAMMA VOOR EEN DISK

Het COPY-kommando werkt onder MSX zeer langzaam wanneer men slechts over één schijveneenheid beschikt. Meestal moet de schijf vele tientallen malen worden verwisseld bij een kopiëeropdracht.

Het volgende programma kopiëert een hele schijf en doet dit met stappen van 16 kilobytes. Hierdoor werkt dit programma veel sneller dan het gewone COPY-kommando.
Het nadeel is dat altijd de gehele schijf wordt gekopiëerd.

Ook in dit programma wordt het DSKI$-kommando gebruikt. Omdat dit geen door Microsoft ondersteund kommando is, kan het zijn dat uw programma niet meteen loopt. In dat geval dient het een en ander te worden versleuteld.

Vergeet niet om voor de zekerheid bij het kopiëren de originele schijf altijd WRITE PROTECT te maken.

```
10 REM ***********************
20 REM *     SNEL KOPIEREN     *
30 REM ***********************
40 REM
50 REM ***********************
60 REM *    HOOFDPROGRAMMA     *
70 REM ***********************
80 REM
90 CLEAR 19000'KAN NIET IN SUBROUTINE
100 GOSUB 190'INITIALISATIE
110 GOSUB 350'OPBOUWEN FILE-LIST
120 IF FP=-1 THEN 170'SCHIJF LEEG
130 GOSUB 560'INLEZEN GEGEVENS
140 IF EE=-1 THEN 170'EINDE
150 GOSUB 820'SCHRIJVEN GEGEVENS
160 GOTO 130'VOLGENDE INLEESGANG
170 GOSUB 1040'AFSLUITEN
180 CLEAR 100:STOP'EINDE
```

```
190 REM *************************
200 REM *      INITIALISATIE    *
210 REM *************************
220 REM
230 DIM F$(111)'VOOR FILELIST
240 DIM F(112)'VOOR LENGTEN FILES
250 EE=0'MERKER EINDE KOPIEERGANG
260 DIM C$(128)'KOPIEERGEBIED
270 FP=0'FILELIST-POINTER OP 0
280 CP=0:PC=0'COPY-POINTER OP 1
290 RP=1:PR=1'RECORD-POINTER OP 0
300 MP=0:PM=0'MEMORY-POINTER OP 0
310 ER=0'MERKER EERSTE KEER
320 WIDTH 40:CLS'BEELD SCHOON
330 PRINT "SNEL ENKEL-DISK KOPIEERPROGRA
MMA   STARK"
340 RETURN'EINDE INITIALISATIE
350 REM *************************
360 REM *   OPBOUWEN FILELIST   *
370 REM *************************
380 REM
390 PRINT "DE FILELIST WORDT NU OPGEBOUW
D"
400 REM ONDERZOEK SECTOREN 5-11
410 REM VOOR SCHIJF-INDEX
420 FOR SE=5 TO 11:K$=DSKI$(1,SE)
430 REM HAAL NU FILELIST OP
440 FOR I=60310! TO 60821! STEP 32
450 REM CONTROLEER JUISTE FILENAME
460 IF PEEK(I)<32 OR PEEK(I)>127 THEN 52
0
470 REM HAAL NU DE BESTANDSNAAM
480 FOR J=I TO I+10
490 F$(FP)=F$(FP)+CHR$(PEEK(J))
500 NEXT J'VOLGENDE LETTER
510 FP=FP+1'VERHOOG FILEPOINTER
```

```
520 NEXT I'VOLGENDE FILENAME
530 NEXT SE'VOLGENDE SECTOR
540 FP=FP-1'CORRIGEER FILELIST POINTER
550 RETURN'FILELIST COMPLEET
560 REM **************************
570 REM * INLEZEN 16KB GEGEVENS *
580 REM **************************
590 REM
600 PRINT:IF ER=1 THEN PRINT "PLAATS ORI
GINELE SCHIJF (RETURN)":ELSE ER=1:GOTO 6
20
610 IF INKEY$<>CHR$(13) THEN 610
620 REM AFVANGEN TWEEDE RETURN
630 IF INKEY$<>"" THEN 630
640 REM WACHTLOOP VOOR SNELLE DRUKKERS
650 FOR I=1 TO 1000:NEXT I
660 ERASE C$:DIM C$(128)'VOOR SNELHEID
670 MO=0'MERKER KANAAL OPEN OF 0
680 IF CP>FP THEN  EE=(MP=0):RETURN
690 IF MO=1 THEN 760
700 F$=F$(CP)'BESTANDSNAAM
710 OPEN F$ AS 1'OPEN BESTAND
720 MO=1
730 FIELD 1,128 AS A$,128 AS B$
740 PRINT:PRINT F$;" WORDT INGELEZEN."
750 ON ERROR GOTO 790
760 GET 1,RP:C$(MP)=A$:C$(MP+1)=B$
770 MP=MP+2:RP=RP+1
780 IF MP=128 THEN MO=0:CLOSE:RETURN ELS
E 690
790 RESUME 800
800 ON ERROR GOTO
810 F(CP)=RP-1:F(CP+1)=1E+20:CLOSE:MO=0:
CP=CP+1:RP=1:GOTO 680
820 REM ***************************
830 REM * SCHRIJVEN 16K GEGEVENS *
```

```
840 REM ************************
850 REM
860 PRINT:PRINT "PLAATS KOPIE SCHIJF (RE
TURN)"
870 IF INKEY$<>CHR$(13) THEN 870
880 REM AFVANGEN TWEEDE RETURN
890 IF INKEY$<>"" THEN 890
900 REM WACHTLOOP VOOR SNELLE DRUKKERS
910 FOR I=1 TO 1000:NEXT I
920 MO=0'MERKER KANAAL OPEN OP 0
930 IF PR>F(PC) THEN CLOSE:MO=0:PC=PC+1:
PR=1:IF PC>FP THEN MP=0:RETURN
940 IF MO=1 THEN 1010
950 F$=F$(PC)'BESTANDSNAAM
960 IF PR=1 THEN OPEN F$ FOR OUTPUT AS 1
:CLOSE'SCHOONMAKEN BESTAND
970 OPEN F$ AS 1'OPEN BESTAND
980 MO=1
990 FIELD 1,128 AS A$,128 AS B$
1000 PRINT:PRINT F$;" WORDT GESCHREVEN."
1010 LSET A$=C$(PM):LSET B$=C$(PM+1):PUT
 1,PR
1020 PM=PM+2:PR=PR+1
1030 IF PM=128 THEN PM=0:MP=0:MO=0:CLOSE
:RETURN ELSE 930
1040 REM ************************
1050 REM *        AFSLUITEN        *
1060 REM ************************
1070 REM
1080 CLOSE: PRINT:PRINT "HET KOPIEREN IS
 TEN EINDE"
1090 RETURN
```

Aantekeningen